68' MICRO JOURNAL

Australia A $ 4.75 New Zealand NZ $ 8.50
Singapore S $ 9.45 Hong Kong H $23.50
Malaysia M $ 9.45 Sweden 30:-SEK

$2.95 USA

Motorola VME-MACINTOSH-S 50 & Other 68XXX Systems
6809 68008 68000 68010 68020 68030
OS-9 The Magazine for Motorola CPU Devices For Over a Decade! A User Contributor Journal FLEX SK*DOS

This Issue:

Reading & Writing MS-DOS/PC-DOS p.17
"C" User Notes
Proposed ANSI C Standard p.12
Mac-Watch p.20
Basically OS-9 p.8

And Lots More!

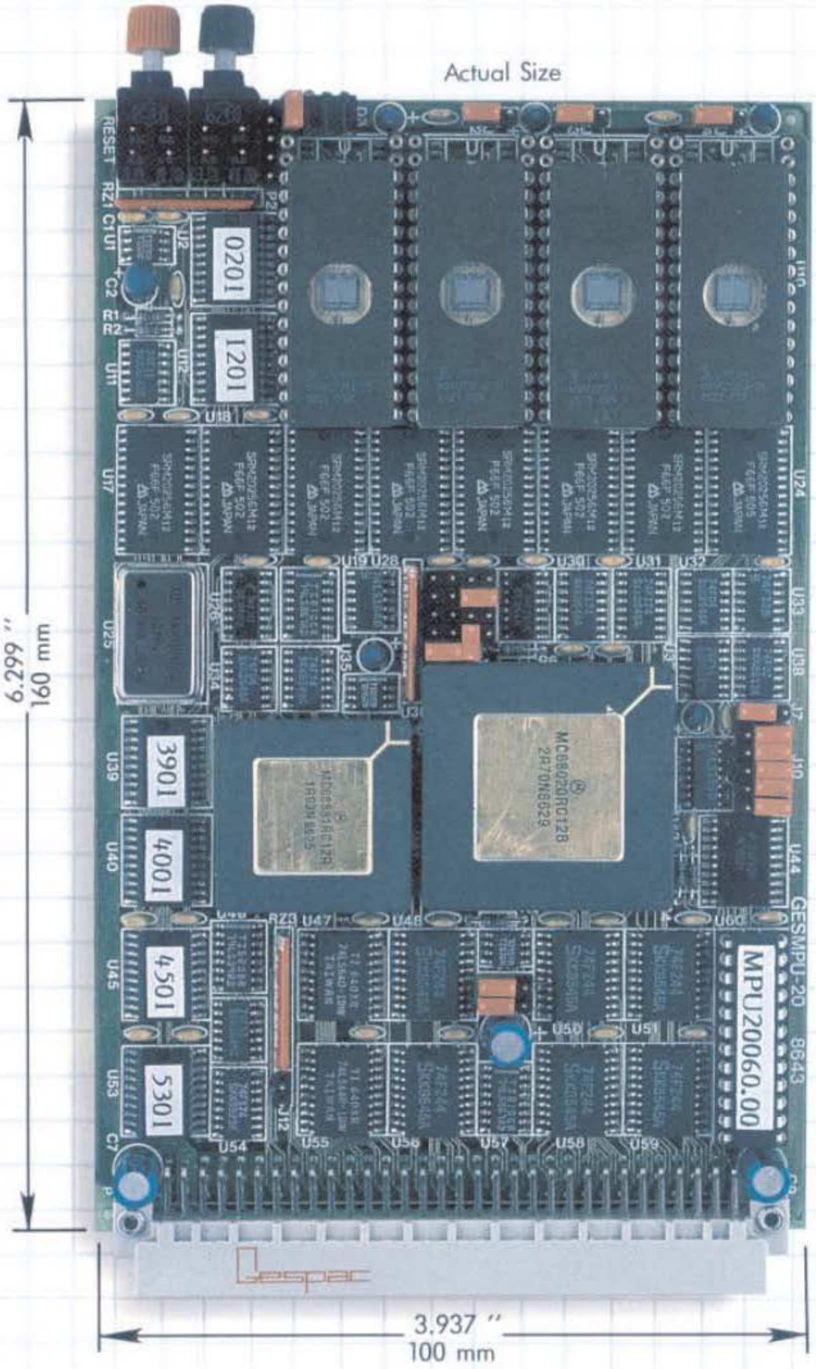
VOLUME IX ISSUE II ● Devoted to the 68XXX User ● February 1987

"Small Computers Doing Big Things"

000422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082

02
0 74470 12810

GESPAC Gives You More Power Per Square Inch.



Here is the size, performance, and cost breakthrough you have been waiting for: The 68020 based GESMPU-20 from GESPAC.

You can now unleash an unprecedented amount of power into your applications. On just 25 square inches, we have squeezed a 12.5 MHz (16 MHz optional) 68020 32-bit microprocessor, a 68881 floating point coprocessor, 4 sockets for up to 512 Kilobytes of EPROM, and up to 512 Kilobytes of zero-wait-states CMOS RAM.

This board is totally expandable through its G-64 bus interface. And GESPAC has the largest variety of inexpensive memory, interfaces, controllers, and transducer cards anywhere. Plus real-time disk operating systems, high level languages, and other software tools. GESPAC has the total solution to your system integration needs.

Best of all, because our boards are small, they cost less. The new GESMPU-20 is priced below $1000 in one hundred piece quantity orders.

So why wait? Contact us today for information on the GESMPU-20 or any of the 150 G-64 bus system components from GESPAC—the leader in single Eurocard microcomputer products worldwide.

Call (602) 962-5559.

Gespac

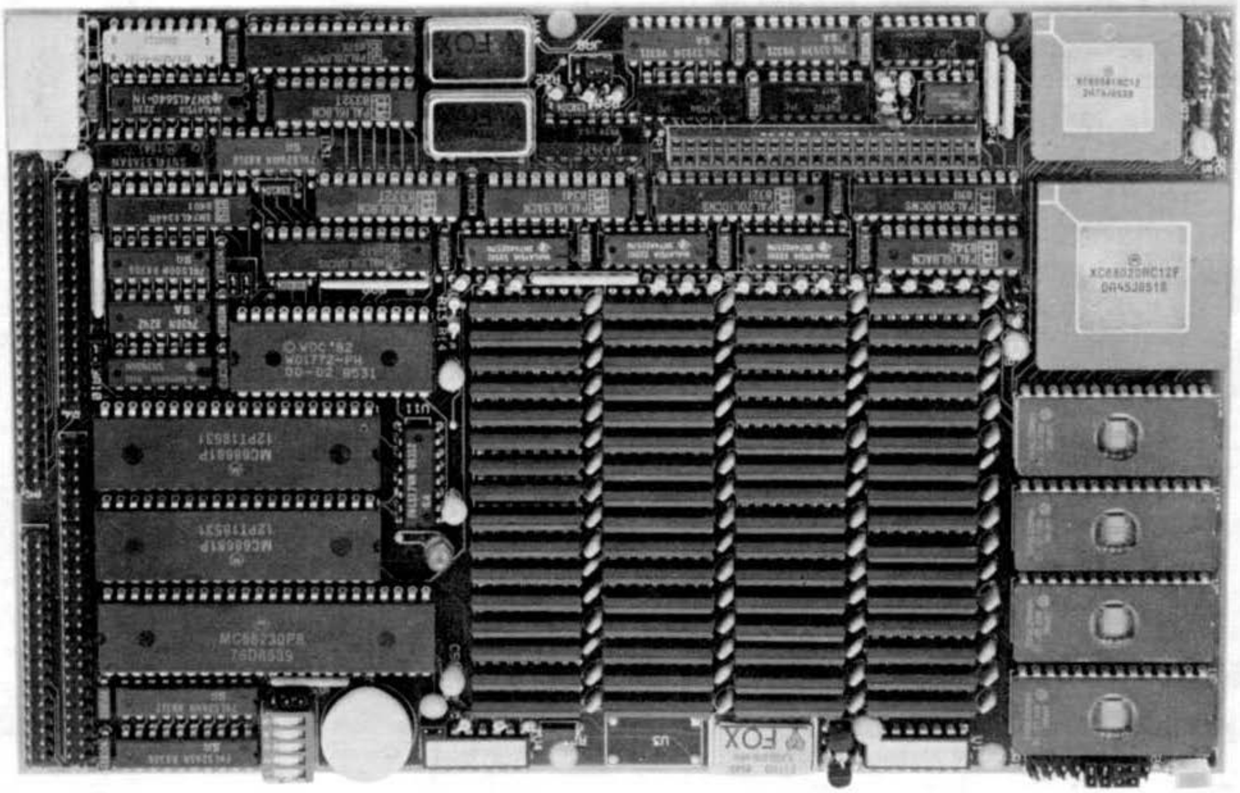
IN USA - CANADA
50A West Hoover Ave.
Mesa, Arizona 85202
Tel. (602) 962-5559
Telex 386575

INTERNATIONAL
3, chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

GMX™ Micro-20
68020 SINGLE—BOARD COMPUTER

Mainframe CPU Performance on a 5.75″ x 8.8″ Board
(benchmark results available on request)

$2565⁰⁰ 12.5 MHz Version
Quantity Discounts Available



Features

- 32-Bit MC68020 Processor (12.5, 16.67, or 20MHZ)
- MC68881 Floating-point coprocessor (optional)
- 2 Megabytes of 32-bit wide, high-speed RAM
- 4 RS-232 Serial I/O Ports (expandable to 36)
- 8-bit Parallel I/O Port ('Centronics' compatible)
- Time-of-Day Clock w/battery backup
- 16-bit I/O Expansion Bus
- Up to 256 Kbytes of 32-bit wide EPROM
- Floppy Disk Controller for two 5¼″ drives
- SASI Intelligent Peripheral Interface (SCSI subset)
- Mounts directly on a 5¼″ Disk Drive
- Optional Boards include Arcnet, Prototyping, I/O Bus adapter, 60 line Parallel I/O, RS-422/485

Software

Included:
- GMX Version of Motorola's 020Bug Debugger with up/download, breakpoint, trace, single-step, and assembler/disassembler capabilities
- Comprehensive Hardware Diagnostics

Optional:

UNIX™ like Multi-user/Multi-tasking Disk Operating Systems
- OS-9/68000™ (Real-time and PROMable)
- UniFLEX™

Programming Languages and Application Software
- BASIC, C, PASCAL, ABSOFT FORTRAN, COBOL and ASSEMBLER
- Spreadsheet, Data Base Management, and Word Processing

COMPLETE EVALUATION SYSTEMS AVAILABLE

GMX inc 1337 W. 37th Place Chicago, IL 60609

(312) 927-5510 • TWX 910-221-4055
State-of-the-Art Computers Since 1975

A Member of the CPI Family

68 Micro Journal

6800 6809 68000 68010 68020

10 Years of Dedication to Motorola Users

Editorial Staff

Publisher:
Don Williams Sr.

Executive Editor:
Larry Williams

Production Manager:
Tom Williams

Administration:
Office Manager:
Mary Robertson
Subscriptions:
Joyce Williams

Contributing & Associate Editors:

Ron Anderson
Ron Voigts
Doug Lurie
David Lewis
Dr. E.M. Bud Pass
Art Weller
Dr. Theo Elbert
& hundreds more of us

Contents

Basically OS-9	8	Voigts
"C" User Notes	12	Pass
Reading & Writing		
MS-DOS/PC DOS	17	Stark
Mac-Watch	20	Review
Reversi	23	Zychlinski
Ramblings	27	DMW
Virtual Terminal	36	Staff
Bit Bucket	37	All of Us
Classifieds	52	

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
CPI
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 - Telex 510 600-6630

Copyright © 1987 Computer Publishing, Inc.
68 Micro Journal is published 12 times a year by Computer Publishing, Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year: $24.50 USA, Canada & Mexico $34.00 *a year. Others add $12.00 a year surface, airmail add $48.00 a year, USA funds!* 2 Years $42.50, 3 Years $64.50 *plus additional postage, for each* additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number and date, as well as a statement that the material is original and the property of the submitting author. Articles submitted should be on diskette, Macintosh, OS-9 or FLEX format. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please.

Please do not *format with spaces any text indents, chart items, etc. (source listings o.k.) WE will edit in ALL formatting*. Text should be flush left column and use ONLY a carriage return to separate paragraphs or other article text items! MacWrite, FLEX TSC, Stylo formatting acceptable.

Letters & Advertising Copy

Letters to the Editor should be original copy, signed! Letters of gripe as well as praise are acceptable. *We reserve the right to reject any letter to the editor or advertising copy material, for any reason we deem advisable.*

Advertising Rates: Commercial please contact 68 Micro Journal advertising department. Classified ads must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word after the first 15. All classifieds must be pre-paid. No classifieds accepted by telephone.

The VME BUS and OS-9:

Ultimate Software for the Ultimate Bus.

Modularity. Flexibility. High Performance. Future growth. These are probably the prime reasons you chose the VME bus. Why not use the same criteria when selecting your system software? That's why you should take a look at Microware's OS-9/68000 Operating System—it's the perfect match for the VME bus.

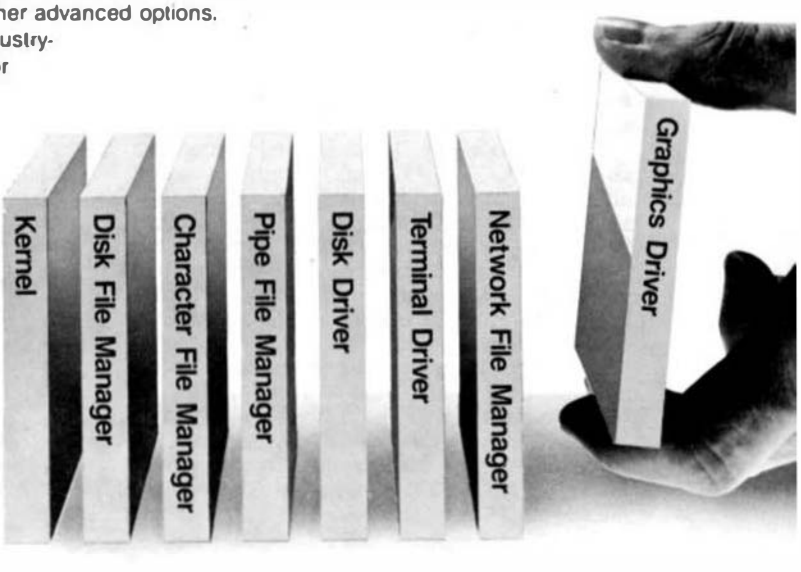
When you're working with VME you must have access to every part of the system. Unlike other operating systems that literally scream KEEP OUT!, OS-9's open architecture invites you to create, adapt, customize and expand. Thanks to its unique modular design, OS-9 naturally fits virtually any system, from simple ROM-based controllers up to large multiuser systems.

And that's just the beginning of the story. OS-9 gives you a complete UNIX-application compatible environment. It is multitasking, real time, and extremely fast. And if you're still not impressed, consider that a complete OS-9 executive and I/O driver package typically fits in less than 24K of RAM or ROM.

Software tools abound for OS-9, including outstanding Microware C, Basic, Fortran, and Pascal compilers. In addition, cross C compilers and cross assemblers are available for VAX systems under Unix or VMS. You can also plug in other advanced options, such as the GSS-DRIVERS™ Virtual Device Interface for industry-standard graphics support, or the OS-9 Network File Manager for high level, hardware-independent networking.

Designed for the most demanding OEM requirements, OS-9's performance and reliability has been proven in an incredible variety of applications. There's nothing like a track record as proof: to date, over 200 OEMs have shipped more than 100,000 OS-9-based systems.

Ask your VME system supplier about OS-9. Or you can install and evaluate OS-9 on your own custom system with a reasonably priced Microware PortPak™. Contact Microware today. We'll send you complete information about OS-9 and a list of quality manufacturers who offer off-the-shelf VME/OS-9 packages.



Modular Hardware Deserves Modular Software

MICROWARE.

Microware Systems Corporation
1866 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

Microware Japan, Ltd.
41-19 Honcho 4-Chome, Funabashi City • Chiba 273,
Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB
S:t Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
Phone: 018-138595
Telex: 76129

Dr. Rudolf Keil, GmbH
Porphystrasse 15
D-6905 Schriesheim
West Germany
Phone: (0 62 03) 67 41
Telex: 465025

Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83-3377
Telex: 828275

Vivaway Ltd.
36-38 John Street
Luton, Bedfordshire, LU1 2JE
United Kingdom
Phone: (0582) 423425
Telex: 825115

Microprocessor Consultants
92 Bynya Road
Palm Beach 2108
NSW Australia
Phone: 02-919-4917

Microdata Soft
97 bis, rue de Colombes
92400 Courbevoie
France
Phone: 1-768-80-80
Telex: 615405

OS-9 is a trademark of Microware and Motorola. PortPak is a trademark of Microware. GSS-Drivers is a trademark of Graphic Software Systems, Inc. VAX and VMS are trademarks of DEC. Unix is a trademark of AT&T.

MUSTANG-020 Super SBC™

DATA-COMP proudly presents the first Under $5000 "SUPER MICRO".

The MUSTANG-020™

CPI
Mustang-020™
A DATA-COMP
Hi-Speed
Product

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. rovisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heav math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COM "total package", consis.ng of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP
Installed Systems World-Wide
OVER 14 YEARS OF DEDICATED QUALITY!

CPI
A Division of
Computer Publishing, Inc.
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 810 600-6630

MUSTANG-020, MUSTANG-08 Benchmarks

All timings by independent consultant

	Time - seconds 32 bit Integer	Register Long
IBM AT 7300 Xenix Sys 3	9.7	no register
AT&T 7300 UNIX PC 68010	7.2	4.3
DEC VAX 11/780 UNIX Berkley 4.2	3.6	3.2
DEC VAX 11/750 " "	5.1	3.2
68000 OS-9 68K 10 Mhz	6.5	4.0
68000 OS-9 68K 8 Mhz	18.0	9.0
MUSTANG-08 68008 OS-9 68K 10 Mhz	9.8	6.3
MUSTANG-020 68020 OS-9 68K 16 Mhz	2.2	0.88
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz	1.8	1.22

```
Main()
    {
        register long i;
        for (i=0; i < 999999; ++i);
    }
```

Estimated MIPS - MUSTANG-020 .... 4.5 MIPS.
Burst to 8 - 10 MIPS: Motorola Specs.

MUSTANG-020™ Software

OS-9

OS-9	$350.00
Basic09	300.00
C Compiler	400.00
Fortran 77	400.00
Microware Pascal	400.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00
Cross Assemblers	50.00
Cmodem w/source	100.00
Disassemblers	100.00

UniFLEX

UniFLEX	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Option & Expansions

8 Port expansion RS-232 (total of 20 serial ports supported)	698.00
Expansion for Motorola I/O Channel Modules	$195.00

** All Expansion boards
All expansion boards for old style cabinet will require the 101 expansion cable. Systems ordered with newer PC type cabinet do not require this cable.

101 Expansion Cable	$39.95

Sculptor+: We are USA distributors for Sculptor+. Call or write for site or multiple system license & discounts, OEM/Dealer.

Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. Save $300.00!!

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quote. Discounts apply after the sale as well.

For a limited time we are offering a $400.00 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for details.

NOTE: UniFLEX is reported to run slower than OS-9 with more than several users on line - Also call or write for information on OS-9 Version 2, soon to be available. A full 68020 OS-9, with 68881 support.

MUSTANG-020™ FEATURES

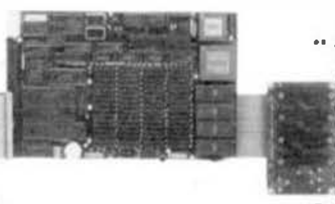
- 12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path processor
- 32-bit wide data and address buses, non-multiplexed
- on chip instruction cache
- object code compatible with all 68XXX family processors
- enhanced instruction set - math co-processor interface
- 68881 math hi-speed floating point co-processor (optional)
  direct extension of full 68020 instruction set
  full support IEEE P754, draft 10.0
  transcendental and other scientific math functions
- 2 Megabyte of SIP RAM (512 x 32 bit organization)
- up to 256K bytes of EPROM (64 x 32 bits)
- 4 Asynchronous serial I/O ports standard
  optional to 20 serial ports
  standard RS-232 interface
  optional network interface
- buffered 8 bit parallel port (1/2 MC68230)
  Centronics type pinout
- expansion connector for additional I/O devices
  16 bit data path
  256 byte address space
  2 interrupt inputs
  clock and control signals
  Motorola I/O Channel Modules
- time of day clock/calendar w/battery backup
- controller for 2, 5 1/4" floppy disk drives
  single or double side, single or double density
  35 to 80 track selectable (48-96 TPI)
- SASI interface
- programmable periodic interrupt generator
  interrupt rate from micro-seconds to seconds
  highly accurate time base (5 PPM)
- 5 bit sense switch, readable by the CPU
- hardware single-step capability
- mounts directly to a standard 5 1/4" disk drive

CPI Mustang-020™ A DATA-COMP Hi-Speed Product

Size 8 15/16 x 5 7/8

These *hi-speed* 68020 systems are presently working at NASA, Atomic Energy Commission, other Government Agencies as well as Universities, Business, Labs, and critical applications centers, Worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a *must!*

MUSTANG-020 System component prices - Effective July 1, 1986
Prices subject to change - call for latest quotes.



MUSTANG-020 (12.50 Mhz)		$2750.00
** Cabinet	(PC or as shown)	$299.95
5"-80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K		$350.00
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Xebec H/D controller		$395.00
Shipping USA UPS		$20.00
Total:		$5059.80

☆☆ **DISCOUNT LIMITED TIME:** Complete System $1061.00

Complete System :

25 Mbyte HD **$3998.80**

85 Mbyte HD **$5248.80**

OPTIONS ADD:

UniFLEX	$90.00	
MC68881 f/p math processor	$275.00	This price subject to increase
16.67 Mhz MC68020	$375.00	Additional MUSTANG systems soon
16.67 Mhz MC68881	$375.00	

Note: Current OS-9 (Ver. 1.2) does not address the MC68881 - Future revisions will. If the 68881 is anticipated in the future, it must be ordered with the system, when originally ordered. UniFLEX does support both the enhanced code of the 68020 and 68881 now.
OPTION BOARDS: ** *Option boards to be installed in Mustang-020 cabinets must be ordered with the extension cable.* The cabinet is too tight for direct plugon. Or specify our new PC type cabinet, with initial order.

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

For Over a Decade!
1,200 Titles +

A Division of

CPI

S.E. Media
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

An Ace of a System in Spades!

MUSTANG-08™

ONE PENNY SALE

NOT **128K,** ***NOT*** **512K**
FULL **768K No Wait RAM**

1¢

C Compile times: OS-9 68K. Hard Disk file. LIST utility source from K&R.

MUSTANG-08	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

The MUSTANG-08™ system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-taskin[illegible] system. All the [illegible]pular 68000 OS-9 software runs. It is [illegible]*d whiz on di*[illegible]ct is: the MUSTANG-08 is faster [illegible] access tha[illegible]e [illegible] 68XXX systems are on memory [illegible]ccess. No[illegible] is fast! And that is [illegible] small part of th[illegible]! See benc[illegible]

Intro[illegible] price of $1,[illegible] *(2-80* [illegible] *flopp*[illegible] *Complete in* [illegible] *style cabinet, he*[illegible]*duty switching power supply, rf by-passing, ready to run, with your choice of OS-9 68K or SK*DOS.* Add $750 for a single floppy/25 megabyte hard disk system. For those that waited, DATA-COMP didn't forget.

1¢ →

Dual 5" Disk System

♠ $1,998.08

25 Megabyte Hard Disk System

♠ $1,998.09

Specifications: System includes OS-9 68K or SK*DOS - Your Choice

CPU	MC68008	10 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	2 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MC146818	Real Time Clock
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Size: 5.75 X 8 inches - bolts directly to a floppy or HD

*

Unlike other 68008 systems there are several significant differe[illegible]*es. The MUSTANG-08 is a full 10 Megahertz system. The* [illegible] *NO wait* [illegible] *MUSTANG type pe*[illegible]*rmance.*

~~$2,748.08~~

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system. [illegible]M dis[illegible] 480K can be easily configured, leaving 288K [illegible]rogr[illegible]em RAM space. The RAM DISK can be [illegible] to [illegible]ze your application requires (system must [illegible]28[illegible] to i[illegible]er requirements). Leaving the [illegible]de[illegible] th[illegible]al [illegible]r program use. Sufficient [illegible]

[illegible] 00 [illegible]ade-in [illegible]es see [illegible]stang-02 [illegible] page #5

SALE

Limited Time!

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68020

```
                               Seconds   32 bit    Register
                                         Integer   Long
Other 68008 8 Mhz OS-9 68K...18.0...9.0
MUSTANG-08 10 Mhz OS-9 68K....9.8...6.3
                 Main()
                   {
C Benchmark Loop          /* int i; */
                          register long i;
                          for (i=0; i < 999999; ++i);
                   }
```

[illegible]LEX [illegible]ark of TSC
OS-9 is [illegible]ark of Microware
MUSTANG-08 is a trademark of CPI
SK*DOS is a trademark of Star-K

Data-Comp Division

CPI

A Decade of Quality Service"
Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC hi-density FLEX 5" - Call for special info.

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL
60139

About a year ago I was wrapping up one project and wondering what the next one would be. About half of my group would be temporarily assigned to a large project involving two other outside laboratories. I was reassigned to the new project. My duties involved designing the electronics for safety interlocks, selecting appropriate test equipment, interfacing analog signals from test setup to the main control room and writing a part of the program to control the power supplies. Up to now my work had included working with the smaller computers. The project called for using the main computer at the test facility, interfacing to sophisticated hardware.

What this meant to my immediate future was working from a terminal with the computer at some remote region. I had access to the facility from any available terminal, as long as, I didn't mind sitting in someone elses office. As an alternate approach I could run a line to my desk and get my own terminal. So I spent an afternoon stringing 70 feet of cable from an RS-232 MUX to my desk. The cable ran along rafters, along ventilation ducts and window sills. Eventually, it found its way to my desk.

The next hurdle was to get a terminal. We had one on order. Considering the speed of our purchasing department, delivery service, and the distributor, we could expect the new terminal in about 6 weeks to 6 months. In otherwords, I needed a terminal. This meant going through surplus and finding what was available. It soon became apparent that surplus equipment meant old, obsolete and no body wanted it. I saw some devices that had no obvious function. Perhaps some time long ago someone, somewhere, had used them. But time and a lack of manuals had doomed them to sitting on shelves collecting dust. Through all of the grime I found a terminal!

??
Do
You
Remember
The
Editor
????

It was simple CRT. It had a standard typewriter key board, a black and white screen, and an RS-232 jack on the back. I can't really complain. I hooked it up, turned on the power and the system asked me for my user name and password. I was connected to the system!

Besides learning the system, I intended to learn it's editor. The editor was an eloquent one that was really a word processor. It had one main requirement. It used a keypad, as many of new terminals have, to invoke all of the editing commands. And that was one thing my terminal didn't have. I contacted the sysop. After all, there must be other users who don't possess fancy terminals. He informed there was a much older editor on the system. It was a line editor. It permitted editing text on a line-by-line basis. One of its strengths was being able to create macros. Hey! I thought that sounds like the editor that came with my OS-9 system. Needless to say, I didn't get to use the line editor, since my new terminal arrived the next day. But it did renew my interest in the editor that Microware supplies with OS-9.

The OS-9 editor is powerful but easy to learn text editor. Some of its features are small size, multiple buffers, and edit macros. The editor occupies only 5K of memory, leaving plenty of space for large text files. The editing of simultaneous files can be done, since it allows creation of 2 or more buffers. And you can create edit macros that let you customize the editor for your personal use.

After using EDIT you'll will find there are a few commands that you'll use more frequently. Here is a list of the ones I find most useful.

^	move to start of text
/	move to end of text
+n	move forward n lines
-n	move backward n lines
CnS1 S2	replace n times S1 with S2
Dn	delete n lines
EnS1	extend n lines with S1
SnS1	search for n occurrences of str
Ln	list n lines
Q	quit

This is a very small list of possible commands that can be used with the OS-9 editor. In general, n is a number; leaving it out of the command line implies 1. If a "*" is used instead of n, then the number is infinity or as close as we can come to it (65535). S1 and S2 are any string that you wish to use. They are always delimited with some character. It doesn't matter what the character is, as long as it is used through out the command line. We'll show this a little later. Also, when any of the above commands are used the edit pointer is left in front of the target line. The exception is "/", which leaves the pointer after the last line.

A editing sequence can start with a line like:

```
EDIT file
```

If 'file' exists, a temporary file called SCRATCH will be created. On termination of an edit session, the original will be deleted and SCRATCH will be renamed. You must have DEL and RENAME in your commands directory, CMDS. One time, I attempted to edit a file that belonged to someone else. It had public read, but not public write. The edit session ended with the original in tack and I had the newer version named SCRATCH. If 'file' does not exist, it will be created. Another variation is:

```
EDIT file1 file2
```

Here 'file2' is created. It is made from 'file1' and whatever you do while editing. The original is left alone.

The editor prints a prompt much like OS-9 does. It prompts with a:

```
E:
```

To enter a line, type a space and then the line you want. Now if you enter something in the first column, the editor assumes it must be sometype of a command. If you meant to enter a line, but started in the first column, the editor will try to interpret it. If it isn't a legitimate command, it responds, "WHAT?"

I usually use only a few commands to move through text. Armed with the +n and -n, I can get anywhere. The ^ and / are convenient also to move to the start and end of text buffer. Many times I will use -* and +* instead of ^ and /. It may be a little slower, though. The +* and -* appears to move through all the lines, while the ^ and / go directly to the start. But the effect is the same. When using these commands the edit pointer is moved to some line. This is an imaginary pointer. If you've used only line number oriented editing this may take a little getting used to. Most find it easy to use with a little practice.

To search for a particular part of text, use S. If you meant to look for "hello there!" enter:

```
E:S/hello there!/
```

The edit pointer will stop directly in front of the first line that matches. Remember the match must be exact. The editor will distinguish between upper and lower case. Also placing a number after the S will cause it to look through that many occurrences of the target text.

```
E:S5"hello there!"
```

will pass over the target text 4 times and stop at the 5th. Notice that I used different type of delimiters to mark the string in each. Again the only requirement is that they be the same on the command line.

The C command is used to change the occurrence of one string to another. So to change the occurrence of "the" for "this " in the next three instances, we could enter:

```
C3!the!this!
```

Here the ! marks strings. Any other character could have been use as long as we are consistent. I many times use C to delete things. A line like:

```
EC3/yes//
```

will remove the next three occurrences of 'yes'. The reverse can also be used.

```
EC//10/
```

will add the number 10 to the beginning of the line.

I included the E command in my most used commands. This one extends line with some target text. When I write assembly language routines, I usually don't put in comments. After I have debugged code, I add the comments. The E command comes in handy here. It extends the line with whatever string I want. I can move through the source code fairly fast, commenting everything I have done.

The L command is useful for listing parts of text. Entering:

```
EL12
```

will list the next 12 lines of text starting at the current edit pointer location. Using L by itself to list one line can be handy, too. It lists the current line and indicates that where the edit pointer is resting.

Finally there is the D command. D will delete lines starting from the location of the edit pointer. To remove 3 lines try:

```
ED3
```

and they're gone. Use the * instead of a number and everything from your current position to the end of the buffer will be deleted.

Commands can be chained together. As long as they start in the first column, then can be entered on one line. What do you think this one does?

```
E:^+3C2/Yes/No/
```

Give up? It moves to the top of the buffer. Moves down three lines. And changes the next three occurrences of 'Yes' to 'No'. Here is one you might want to think twice about:

```
E:^D*
```

When finished with an editing session, enter Q. The buffer will be written out to the file. An old file will be deleted if necessary and the SCRATCH file be renamed to the new name. If things should go awry, and you find that you are left with s SCRATCH file, you may have to to do some deleting and renaming on your own.

This is only a small smattering of what you can do with the OS-9 editor. There are so many things that I have not covered. There are 34 commands and 26 pseudo macros in the OS-9 Macro Editor. I have only covered a few of the commands this month. You can also create you won macro commands. There is enough here to get you going. In future columns I will talk more about the other commands, the pseudo macros and writing your own macros.

A LETTER AND A PROGRAM

Part of the inspiration for this month's column comes from Phil Chadwick, New Hope, Pa. He uses the OS-9 Editor for programs, as well as, editing text. I can identify with him. Some of the earlier columns went through their final editing with the OS-9 Editor. Phil writes about a problem. Briefly here it is.

He finds while in the editor, after making numerous changes and deletions, it hangs up. He gets a blank line and can't go any further. He tries to delete lines of text and they don't delete. Using a / to move to the end of text doesn't work. The editor stops at the same bad point. Using a D* will not delete to the end of text either.

This problem is a puzzler. I have not been able to duplicate it. It is nearly impossible to trouble shoot something, if it can't be reproduced. I have had some ideas. Since the editor the loads and runs, I've ruled out a damage to the program. The CRC checking would have caught it immediately. Possibly a bad RAM chip could be the culprit. I had a problem of this type a few years back. My word processor would run fine and then after everything warmed up, the text would look like garbage. I ruled out this since Phil doesn't report problems with other programs.

My best solution at this point is the "old hidden character" problem. For whatever reasons, somehow an odd character gets into the ascii file. When loaded into the editor, it signal some message to the editor and creates the problem. One time I received a transmission via modem. I captured it to disk. Later I tried to edit it. No luck! The problem was it had a $0C in it. That happens to be the signal to clear my screen and home the cursor. The net result was every time I edited the file, the screen would clear. You can't edit what you can't see. I eventually created a little program that read the file, and outputted only the printable characters.

If you have had a similar problem with the editor let me know. If you have found a cause and solution, better yet! You can write to me care of the magazine or directly to my address at the beginning of the column. If you have some problem or question drop me line. Include a S.A.S.E. If I can't find an adequate solution, we'll pass it along to the readers. Who knows, someone else may have had the same problem. When you do write, be clear and precise. If you can list step-by-step how to lead up to the problem, do so.

If you think you have a a superfluous character in a file, try this little program. It is written in C, and should be relatively easy to implement. I added line numbers to make it easier to follow. Remember they are not in the original code. Basically the program passes only printable characters to the standard output path and $0D which marks the end of line. Most of the library calls here are standard ones. The only exception is PRERR() which prints the error to the path indicated. I used path #2 since this is the standard error path. I use 'errno' as the error. This variable is found in cstart.a, the start module for C programs compiled with the Microware C Compiler. To use strip, simply enter:

STRIP filename

You can redirect the output to your printer or another file. I use this with files I received by phone via a modem. I am sure you'll find your own applications. It is a handy little program to have around. And may sometime prove useful.

```
        LISTING
         1 /* Program: STRIP
         2     By: Ron Voigts
         3     Date: 16-OCT-86
         4     To Compile: CC1 STRIP.C
         5     Usage: STRIP FILE >NEWFILE
         6
         7        This program will pass only
printable and
         8     EOL characters.  It inputs from a
file and
         9     prints to the standard output.
        */
        10
        11 #include <stdio.h>
        12
        13 main(argc,argv)
        14 int argc;
        15 char *argv[];
        16 {
        17    register int i;
        18    FILE *fp, *fopen();
        19    int c;
        20    for (i=1; i<argc; i++) {
         21                                         if
((fp=fopen(argv[i],"r"))==NULL)
        22          prerr(2,errno);
        23       while ((c=getc(fp))!=EOF)
        24            if ((c>=32 && c<=126) ||
(c=='\n'))
        25             putchar(c);
        26       fclose(fp);
        27    }
        28 }
```

EOF

The C Programmers Reference Source. Always Right On Target!

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter continues the discussion of the proposed ANSI C standard and the discussion of common problem areas in the use of the C language and its libraries.

PROPOSED ANSI C STANDARD

The header file "setjmp.h" declares two functions (setjmp() and longjmp()) and one structure (jmp_buf). These functions and type provide a facility for performing the equivalent of a "long jump" operation by storing and restoring the machine states as required to recreate a calling environment. The facility is limited and potentially dangerous in some implementations.

These functions are as follows:

```
int setjmp(jmp_buf env);
   saves callin genvironmen
   in env, then returns 0
void longjmp(jmp_buf env,
        int val);
   restores calling environment
   saved by setjmp, returns val
   from calling point of setjmp
```

The header file "signal.h" declares two functions and several macros for specifying the handling of various classes of conditions (signals) which may be encountered during program execution.

The minimum set of macros for a given conforming implementation is as follows:

```
SIG_IGN ignore signal
SIG_DFL apply default handling
SIG_ERR error return code
SIGABRT abort signal
SIGFPE  floating-point exception
SIGILL  invalid function image
SIGINT  interrupt request
SIGSEGV invalid access to data
SIGTERM teminaion request
```

The functions are defined as follows:

```
void (*signal(int sig,
    void(*func)()))();
   specifies signal processing
int kill(int pid, int sig);
   sends signal sig to program
   pid and returns zero if ok
```

The header file "stdarg.h" declares a structure (va_list) and several macros for establishing and processing an argument list the contents of which are not assumed known to the processing functions until run time.

The macros are as follows:

```
void va_start(va_list ap, parmN);
   initializes ap for processing
   list with last parameter parmN
type va_arg(va_list ap, type);
   returns type and value of
   next argument of va_list
void va_end(va_list ap);
   concludes processing parameters
```

The header file "stdio.h" declares a structure, macros, and functions for performing input and output operations.

The structure FILE provides the common area used by many of the macros and functions in performing and controlling the input/output operations.

The minimum set of macros is as follows:

```
BUFSIZE
   size of stream buffer
EOF
   end of file
L_tmpnam
   length of temporary file name
SEEK_CUR
   seek relative to current
   location
SEEK_END
```

seek relative to end of file
SEEK_SET
seek relative to top of file
SYS_OPEN
maximum number of open files (must be at least eight)
TMP_MAX
maximum number of temporary file names
stderr
FILE pointer to standard error file
stdin
FILE pointer to standard input file
stdout
FILE pointer to standard output file

The minimum set of functions is as follows:

```
int remove(const char *path);
   removes file with name pointed
   to by path
int rename(const char *old,
        const char *new);
   renames file with name pointed
   to by old to name pointed
   to by new
FILE *tmpfile(void);
   returns pointer to new
   temporary file
char *tmpnam(char *s);
   creates temporary file name
int fclose(FILE *stream);
   flushes and closes stream
int fflush(FILE *stream);
   flushes stream
FILE *fopen(const char *path,
        const char *mode);
   attempts to open file with
   name pointed to by path and
   mode pointed to by mode,
   returns file pointer
FILE *freopen(const char *path,
          const char *mode,
          FILE *stream);
   closes stream, attempts to
   open file as above
void setbuf(FILE *stream,
        char *buffer);
   changes buffering on stream
int fprintf(FILE *stream,
        const char *format,
        ...);
   formats data from parameter
   list according to string
   pointed to by format and
   outputs it to stream,
   returns number of characters
int fscanf(FILE *stream,
        const char *format,
        ...);
   formats data from stream
   according to string pointed
   to by format and places it
   into pointers in parameter
   list
int printf(const char *format,
        ...);
   fprintf(stdout, format, ...)
int scanf(const char *format,
        ...);
   fscanf(stdin, format, ...)
int sprintf(char *string,
        const char *format,
        ...);
   like fprintf except data is
   placed into string pointed
   to by string
int sscanf(char *string,
        const char *format,
        ...);
   like fscanf except data is
   taken from string pointed
   to by string
int vfprintf(FILE *stream,
        const char *format,
        va_list arg);
   like fprintf except argument
   list is replaced by variable
   argument list pointer arg
int vprintf(const char *format,
        va_list arg);
   vfprintf(stdout, format, arg)
int vsprintf(char *string,
        const char *format,
        va_list arg);
   like sprintf except argument
   list is replaced by variable
   argument list pointer arg
int fgetc(FILE *stream);
   attempts to obtain the next
   character from the input
   stream and returns either
   the value of the character
   converted to type int or
   returns EOF
char *fgets(char *string, int n,
        FILE *stream);
   places characters into area
   pointed to by string until
   (n-1) characters have been
   placed, EOF is encountered,
   or a new-line is found
int fputc(int c, FILE *stream);
   writes character c to stream
int fputs(const char *string,
        FILE *stream);
    writes characters pointed to
    by string to stream
int getc(FILE *stream);
    like fgetc except may be
    implemented as a macro
int getchar(void)
    getc(stdin)
char *gets(char *string);
    reads characters from stdin
    and places them into area
    pointed to by string until
    EOF is encountered or a
    new-line is found
int putc(int c, FILE *stream);
    like fputc except may be
    implemented as a macro
int putchar(int c);
    putc(stdout)
int puts(const char *string);
    writes characters pointed to
    by string, then new-line,
    to stdout
int ungetc(int c, FILE *stream);
    pushes character c back into
    input stream
int fread(char *ptr, size_t size,
        int nelem,
        FILE *stream);
    reads into area pointed to by
    ptr up to (size * nelem)
    bytes from stream, returns
    number of elements (groups
    of size) actually read
int fwrite(char *ptr, size_t size,
        int nelem,
        FILE *stream);
    writes from area pointed to by
    ptr up to (size * nelem)
    bytes into stream, returns
    number of elements (groups
    of size) actually written
int fseek(FILE *stream,
        long offset,
        int ptrname);
    sets file pointer for stream
    as signed offset from
    indicated position in file
long ftell(FILE *stream);
    returns current value of
    file pointer for stream
void rewind(FILE *stream);
    fseek(stream, 0L, SEEK_SET)
void clearerr(FILE *stream);
    resets end of file and error
    indicators for stream
int feof(FILE *stream);
    returns non-zero if end of
    file is set for stream
int ferror(FILE *stream);
    returns error indicator
    for stream
char *perror(const char *string);
    maps error number in errno
    into an error message,
    optionally outputs message
    to stderr, and returns a
    pointer to the message
```

C PROBLEM

The previous C problem continued an investigation of the implementation dependencies with respect to data type length and alignment considerations.

The simplest defensive measure which may be taken is to place the longer and more complex declarations in a structure before the shorter and simpler declarations. This may produce the best arrangement in many cases.

Another technique which may be used if only a limited set of different implementations is to be supported is to declare an optimum structure for each case, then use "#ifdef" directives to select the appropriate ones.

The C language supports the following primitive data types:

character,
integer,
floating-point

and a large number of derived and composite data types. Since the char data type is really a subset of the int data type, it probably could be considered non-primitive if there were a data type equivalent to "char int".

What is missing, as far as compatibility and portability is concerned, is the ability to specify the characteristics of the primitive data types, at least beyond signed or unsigned. It is probably beyond the capability of current C compilers to allow the specification of the number of bits in a data type char, but it should not be beyond their capabilities to allow the specification of the the type of alignment, the number of char units or bits or digits in a longer data type, and even the form of arithmetic (binary or decimal). Other languages such as COBOL, FORTRAN, and PASCAL allow specification of some of these parameters.

EXAMPLE C PROGRAM

Following is this month's example C program; it completes the B+ tree program started in an earlier chapter.

```
/* a = underflow page, c = ancestor page */
underflow(c, a, s, h)
REF c, a;
```

```
int     s, *h;
{
    REF b;
    int    i, k, mb, mc;

    if (c == root && c->type.indexp.m == 1 &&
        a->page_type == LEAF)
         return; /* only root left in index */
    if (a->page_type == LEAF)
    {
        mc = c->type.indexp.m;
        /* h = true, a->m = N-1 */
        if (s < mc)
        { /* b = page to the right of a */
            s += 1;
            b = c->type.indexp.e[s].p;
            mb = b->type.leafp.k;
            k = (mb - L + 1) / 2;
            /* k = no. of items available
               on adjacant page b */
            if (k > 0)
            { /* move k items from b to a */
                for (i = 1; i <= k; i++)
                {
                    a->type.leafp.d[i+L-1].key =
                        b->type.leafp.d[i].key;
                    a->type.leafp.d[i+L-1].count =
                        b->type.leafp.d[i].count;
                }
                c->type.indexp.e[s].key =
                    b->type.leafp.d[k+1].key;
                mb -= k;
                for (i = 1; i <= mb; i++)
                {
                    b->type.leafp.d[i].key =
                        b->type.leafp.d[i+k].key;
                    b->type.leafp.d[i].count =
                        b->type.leafp.d[i+k].count;
                }
                b->type.leafp.k = mb;
                a->type.leafp.k = L - 1 + k;
                *h = 0;
            }
            else
            { /* merge pages a and b */
                for (i = 1; i <= L; i++)
                {
                    a->type.leafp.d[i+L-1].key =
                        b->type.leafp.d[i].key;
                    a->type.leafp.d[i+L-1].count =
                        b->type.leafp.d[i].count;
                }
                for (i = s; i <= mc - 1; i++)
                {
                    c->type.indexp.e[i].key =
                        c->type.indexp.e[i+1].key;
                    c->type.indexp.e[i].p =
                        c->type.indexp.e[i+1].p;
                }
                a->type.leafp.k = LL - 1;
                c->type.indexp.m = mc - 1;
                *h = c->type.indexp.m < N;
            }
        }
        else
```

```
        { /* b = page to the left of a */
            if (s == 1)
                b = c->type.indexp.p0;
            else
                b = c->type.indexp.e[s-1].p;
            mb = b->type.leafp.k;
            k = (mb - L + 1)
                / 2;
            if (k > 0)
            { /* move k items from page b to a */
                for (i = L - 1; i >= 1; i--)
                {
                    a->type.leafp.d[i+k].key =
                        a->type.leafp.d[i].key;
                    a->type.leafp.d[i+k].count =
                        a->type.leafp.d[i].count;
                }
                mb = mb - k;
                for (i = k; i >= 1; i--)
                {
                    a->type.leafp.d[i].key =
                        b->type.leafp.d[i+mb].key;
                    a->type.leafp.d[i].count =
                        b->type.leafp.d[i+mb].count;
                }
                c->type.indexp.e[s].key =
                    a->type.leafp.d[1].key;
                b->type.leafp.k = mb;
                a->type.leafp.k = L - 1 + k;
                *h = 0;
            }
            else
            { /* merge pages a and b */
                for (i = 1; i <= L - 1; i++)
                {
                    b->type.leafp.d[i+mb].key =
                        a->type.leafp.d[i].key;
                    b->type.leafp.d[i+mb].count =
                        a->type.leafp.d[i].count;
                }
                b->type.leafp.k = LL - 1;
                c->type.indexp.m = mc - 1;
                *h = c->type.indexp.m < N;
            }
        }
    }
    else
    { /* index pages */
        mc = c->type.indexp.m;
        /* h = true, a->m = N-1 */
        if (s < mc)
        { /* b = page to the right of a */
            s += 1;
            b = c->type.indexp.e[s].p;
            mb = b->type.indexp.m;
            k = (mb - N + 1) / 2;
            /* k = no. of items available
               on adjacant page b */
            a->type.indexp.e[N].key =
                c->type.indexp.e[s].key;
            a->type.indexp.e[N].p =
                b->type.indexp.p0;
            if (k > 0)
            { /* move k items from b to a */
                for (i = 1; i <= k - 1; i++)
            {
                a->type.indexp.e[i+N].key =
                    b->type.indexp.e[i].key;
                a->type.indexp.e[i+N].p =
                    b->type.indexp.e[i].p;
            }
            c->type.indexp.e[s].key =
                b->type.indexp.e[k].key;
            b->type.indexp.p0 =
                b->type.indexp.e[k].p;
            mb -= k;
            for (i = 1; i <= mb; i++)
            {
                b->type.indexp.e[i].key =
                    b->type.indexp.e[i+k].key;
                b->type.indexp.e[i].p =
                    b->type.indexp.e[i+k].p;
            }
            b->type.indexp.m = mb;
            a->type.indexp.m = N - 1 + k;
            *h = 0;
        }
        else
        {   /* merge pages a and b */
            for (i = 1; i <= N; i++)
            {
                a->type.indexp.e[i+N].key =
                    b->type.indexp.e[i].key;
                a->type.indexp.e[i+N].p =
                    b->type.indexp.e[i].p;
            }
            for (i = s; i <= mc - 1; i++)
            {
                c->type.indexp.e[i].key =
                    c->type.indexp.e[i+1].key;
                c->type.indexp.e[i].p =
                    c->type.indexp.e[i+1].p;
            }
            a->type.indexp.m = NN;
            c->type.indexp.m = mc - 1;
            *h = c->type.indexp.m < N;
        }
    }
    else
    {   /* b = page to the left of a */
        if (s == 1)
            b = c->type.indexp.p0;
        else
            b = c->type.indexp.e[s-1].p;
        mb = b->type.indexp.m + 1;
        k = (mb - N) / 2;
        if (k > 0)
        {   /* move k items from page b to a */
            for (i = N - 1; i >= 1; i--)
            {
                a->type.indexp.e[i+k].key =
                    a->type.indexp.e[i].key;
                a->type.indexp.e[i+k].p =
                    a->type.indexp.e[i].p;
            }
            a->type.indexp.e[k].key =
                c->type.indexp.e[s].key;
            a->type.indexp.e[k].p =
                a->type.indexp.p0;
            mb = mb - k;
            for (i = k - 1; i >= 1; i--)
            {
                a->type.indexp.e[i].key =
                    b->type.indexp.e[i+mb].key;
                a->type.indexp.e[i].p =
                    b->type.indexp.e[i+mb].p;
            }
            a->type.indexp.p0 =
                b->type.indexp.e[mb].p;
            c->type.indexp.e[s].key =
                b->type.indexp.e[mb].key;
            b->type.indexp.m = mb - 1;
            a->type.indexp.m = N - 1 + k;
            *h = 0;
        }
        else
        {   /* merge pages a and b */
            b->type.indexp.e[mb].key =
                c->type.indexp.e[s].key;
            b->type.indexp.e[mb].p =
                a->type.indexp.p0;
            for (i = 1; i <= N - 1; i++)
            {
                b->type.indexp.e[i+mb].key =
                    a->type.indexp.e[i].key;
                b->type.indexp.e[i+mb].p =
                    a->type.indexp.e[i].p;
            }
            b->type.indexp.m = NN;
            c->type.indexp.m = mc - 1;
            *h = c->type.indexp.m < N;
        }
    }
  }
}


printtree(p, l)
REF p;
int    l;
{
    int    i;

    if (p->page_type != LEAF)
    {
        for (i = 1; i <= l; i++)
            printf("   ");
        for (i = 1; i <= p->type.indexp.m; i++)
            printf("%4d", p->type.indexp.e[i].key);
            printf("\n");
            printtree(p->type.indexp.p0, l + 1);
            for (i = 1; i <= p->type.indexp.m; i++)
                printtree(p->type.indexp.e[i].p, l + 1);
    }
    else
    {
        for (i = 1; i <= l; i++)
            printf("   ");
        for (i = 1; i <= p->type.leafp.k; i++)
            printf("%4d", p->type.leafp.d[i].key);
        printf("\n");
    }
}
```

EOF

Reading & Writing MS-DOS/PC-DOS Under SK-DOS

by: Peter A. Stark
Star-K Software Systems Corp.
P.O. Box 209, Mt. Kisco, NY 10549

This article describes a very simple method for transferring text files between IBM-compatible systems running MS-DOS and PC-DOS, and 68K systems running SK*DOS/68K. This method also works for some 6809 systems running SK*DOS/6809 and Flex. (And let's satisfy all the legal beagles by stating that MS-DOS is a trademark of Microsoft, PC-DOS is a trademark of IBM, Flex is a trademark of Technical Systems Consultants, and SK*DOS is a trademark of Star-K Software Systems Corp.)

There are quite a few differences between MS/PC-DOS disks and SK*DOS or Flex disks:

1. The directory and file structure is totally different on an MS/PC-DOS disk. But the simplified method in this article totally ignores the directory, and so this does not matter to us at all. Not having to read or understand the MS/PC-DOS disk's directory structure tremendously simplifies the task.

2. MS/PC-DOS disks are written entirely in double density, including track 0. SK*DOS/68K has no trouble reading a double-density track 0, whereas 6809 systems can not. But this doesn't matter because we don't read or write the MS/PC-DOS disk directory on track 0.

3. MS/PC-DOS disks use 512-byte sectors, whereas SK*DOS and Flex disks use 256-byte sectors. The sector length is, however, encoded directly in the sector format, so the disk controller on a 6809 or 68K system will generally read or write the longer sector automatically as long as we provide a longer File Control Block (FCB) to hold the extra data. Thus the FCB on a 6809 system must be 576 bytes long (256 bytes longer than the standard 320 bytes), and the FCB with SK*DOS/68K must be 608 bytes.

4. MS/PC-DOS disks use 9 sectors per side whereas SK*DOS and Flex use 18 sectors in double density. This really only causes a problem in sector numbering on side 2, and so we simply require the MS/PC-DOS disk to be formatted as single-sided.

The only remaining problem is caused by the fact that MS/PC-DOS disks are totally double-density. Unfortunately, there are two methods of reading and writing 6809 double-density disks, depending on the type of disk controller. SWTP and similar double-density controllers use one type of disk format, whereas Gimix and similar controllers use the other format, and they cannot read each other's double-density disks. MS/PC-DOS disks use the same format as the Gimix-type controller, so Gimix controllers (and others using the same format such as the Elektra) will be able to read MS/PC-DOS disks, whereas SWTP cannot. This is a basic limitation of the hardware and there is no software method of getting around it. (No problem with SK*DOS/68K, since all systems running it use the Gimix method.)

To transfer a disk between an MS/PC-DOS system and a SK*DOS system, first format the disk as single-sided using the MS/PC-DOS command

```
FORMAT A:/1
```

(which assumes that the disk is in drive A:).

To transfer a text file from the MS/PC-DOS system to your SK*DOS system, simply copy it to this disk and then convert it with FROMSDOS (described below). When MS/PC-DOS writes a file to a freshly-formatted single-sided disk, it starts writing it on track 1 sector 1 and continues without any breaks until finished, at which time it puts a control-Z marker at the end. Since the file is always written in this manner, FROMSDOS knows exactly where the file is and hence has no need to read the directory!

To transfer a file back from your SK*DOS system to MS/PC-DOS without having to write the directory on the SK*DOS system, we let MS/PC-DOS do the job by running the following Basic program on your MS/PC-DOS system (it takes a minute or two):

```
10 OPEN "A:TEXT.TXT" FOR OUTPUT AS 1  20 PRINT #1, "GARBAGE"  30 GOTO 20
```

This writes a file called TEXT.TXT on the entire disk. The file is full of 'garbage', which is unimportant; the important thing is that it puts an entry for this file into the disk's directory. Now take the disk to your SK*DOS system and run TOMSDOS to copy the desired file to the disk, replacing the original file contents. Note that the original TEXT.TXT file occupied the entire disk, whereas the new file is probably shorter. No problem, because TOMSDOS puts a control-Z marker at the end of it to tell MS/PC-DOS where the file ends.

The disk can now be read on your MS/PC-DOS system. The TYPE program and most text editors will only read up to the control-Z, but if you wish to copy the file you must use the /A option of MS/PC-DOS's COPY to tell it to copy only up to the control-Z, rather than the entire disk. For example, to copy the file to a hard disk, the command would be

```
COPY /A A:TEXT.TXT C:
```

The source code for FROMSDOS and TOMSDOS is given below; The 68K code is shown since all SK*DOS/68K users can run it, whereas 6809 SK*DOS and Flex users can only use it if they have an appropriate disk controller. The translation to 6809 code is fairly easy, but feel free to send me a disk if you would like it done for you.

The syntax for using FROMSDOS is

```
FROMSDOS <MS/PC-DOS drive number>
<SK*DOS file spec>
```

For example, to copy a file from an MS/PC-DOS disk in drive 1 to FILE.TXT in drive 0, the command would be

```
FROMSDOS 1 FILE
```

The SK*DOS file specification defaults to the working drive and a .TXT extension, unless specified otherwise, so the file goes to 0.FILE.TXT if drive 0 is the working drive.

The syntax for using TOMSDOS is

```
TOMSDOS <SK*DOS file spec>
<MS/PC-DOS drive number>
```

For example,

```
TOMSDOS FILE 1
```

would copy 0.FILE.TXT to a previously formatted MS/PC-DOS disk in drive 1. To avoid potential problems, TOMSDOS tries to read track 0 sector 6 of the MS/PC-DOS disk first. If this generates an error (on a 6809 system), or else if bytes 5 through 8 of the sector contain zeroes, then TOMSDOS prints a warning message that the disk may not be a MS/PC-DOS disk and asks whether to proceed. This is to avoid the possibility of clobbering an SK*DOS disk by writing an MS/PC-DOS file on top of it.

The 68K source code for the two programs follows.

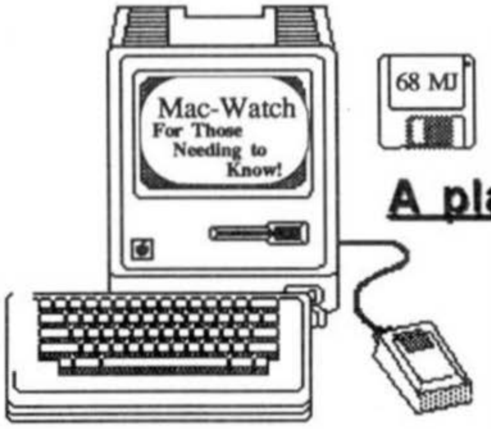
Note: this article and source codes are on Reader Service Disk #31 - see advertising page 62 - this issue of 68 Micro Journal.

```
*****************************************************************
* FROMSDOS UTILITY FOR SK*DOS /68K                              *
* THIS UTILITY TRANSFERS A FILE FROM MSDOS FORMAT               *
* TO SK*DOS/68K. TRANSLATION TO 6809 CODE IS STRAIGHTFORWARD.   *
*****************************************************************

* COPYRIGHT (C) 1986 BY PETER A. STARK

* EQUATES TO SK*DOS

DEFEXT  EQU  $A024      Default extension
FCBCSE  EQU  35         Current sector in buffer
FCBCTR  EQU  34         Current track in buffer
FCBDAT  EQU  96         Beginning of data buffer (256 bytes)
FCBDRV  EQU  3          Logical Drive number
FCLOSE  EQU  $A008      Close file
FOPENW  EQU  $A006      Open a file for write
FWRITE  EQU  $A002      Write the next byte to the file
GETNAM  EQU  $A023      Get file name into FCB
GETNXT  EQU  $A02D      Get next character from buffer
```

```
PERROR  EQU  $A037      Print error code
PSTRNG  EQU  $A035      Print CR/LF and string
SREAD   EQU  $A01C      Read a single sector
WARMST  EQU  $A01E      Warm start

FRMSDO    BRA.S START                 GO TO START

          DC.W $0100                  VERSION NUMBER

* START OF ACTUAL PROGRAM

START     DC GETNXT                   GET INPUT DRIVE NUMBER
          SUB.B #$30,D5               CONVERT FROM ASCII
          CMP.B #3,D5                 CHECK IF VALID
          BHI.L HELP                  PRINT HELP MESSAGE ON ERROR
          LEA INFCB(PC),A4
          MOVE.B D5,FCBDRV(A4)        STORE INTO INPUT FCB
          MOVE.W #$0100,FCBCTR(A4)    POINT BEFORE TRACK 1 SECTOR 1
          LEA INFCB+FCBDAT+512(PC),A1 POINT PAST INPUT FCB DATA AREA
          MOVE.L A1,A2                DITTO

          MOVE.L A6,A4                POINT TO SYSTEM FCB
          DC GETNAM                   GET FILE SPEC INTO FCB
          BCS.L HELP                  PRINT HELP MESSAGE ON ERROR

* FILE SPEC WAS OK; DEFAULT TO .TXT
          MOVE.B #1,D4                DEFAULT EXTENSION CODE
          DC DEFEXT                   DEFAULT TO .TXT

* NOW ACTUALLY OPEN THE OUTPUT FILE
          MOVE.L A6,A4
          DC FOPENW                   GO OPEN THE FILE FOR WRITING
          BNE.S ERROR                 IF NOT ZERO (E - 0)

* MAIN LOOP TO READ AND OUTPUT EACH CHARACTER
MAIN      BSR.S GETMSD                GO GET MSDOS CHARACTER
          CMP.B #$0A,D4               LF?
          BEQ.S MAIN                  YES, IGNORE IT
          CMP.B #$1A,D4               EOF?
          BEQ.S CLOSE                 YES, CLOSE SK*DOS FILE AND QUIT

          MOVE.L A6,A4                POINT TO SYSTEM FCB
          DC FWRITE                   WRITE NEXT CHARACTER TO SK*DOS FILE
          BNE.S ERROR                 QUIT ON ERROR
          BRA.S MAIN                  ELSE DO NEXT

* GETMSD - GET CHARACTER FROM MSDOS FILE

GETMSD    CMP.L A1,A2                 PAST END OF FCB?
          BNE.S NEXT                  NO, READ NEXT CHARACTER
          LEA INFCB(PC),A4            POINT TO INPUT FCB
          ADD.B #1,FCBCSE(A4)         GO TO NEXT SECTOR
          MOVE.W FCBCTR(A4),D7        TRACK AND SECTOR
          CMP.B #10,D7                PAST SECTOR 9?
          BNE.S RDNEXT                NO, OK TO EXIT
          ADD.W #$0100,D7             YES, SO NEXT TRACK
          MOVE.B #1,D7                AND SECTOR 1
          MOVE.W D7,FCBCTR(A4)
          CMP.W #$2801,D7             PAST TRACK $27?
          BNE.S RDNEXT                NO, OK TO CONTINUE
          LEA ENTMSG(PC),A4
          DC PSTRNG                   PRINT "AT END OF MSDOS DISK
          BRA.S CLOSE                 AND QUIT
RDNEXT    DC SREAD                    GO READ NEXT SECTOR OF MSDOS DISK
          BNE.S ERROR
          LEA INFCB+FCBDAT(PC),A1     POINT TO DATA AREAA
NEXT      MOVE.B (A1)+,D4             GET NEXT CHARACTER
          RTS                         AND EXIT

* HELP ROUTINE

HELP      LEA HLPMSG(PC),A4
          DC PSTRNG                   PRINT HELP MESSAGE
          DC WARMST

* ERROR ROUTINE

ERROR     DC PERROR                   PRINT ERROR CODE
CLOSE     MOVE.L A6,A4                POINT TO FCB
          DC FCLOSE                   GO CLOSE THE FILE
          BEQ.S QUIT
          DC PERROR                   IF THERE IS AN ERROR
QUIT      DC WARMST                   AND RETURN TO SK*DOS

* TEXT STRINGS

HLPMSG    DC.B 'FRMSDOS reads a text file from a specially-formatted'
          DC.B $0D,$0A
          DC.B 'MSDOS disk. The correct syntax is'
          DC.B $0D,$0A
          DC.B '  FRMSDOS <MSDOS drive number> <SK*DOS file spec> ',4
DDRMSG    DC.B 'INVALID INPUT DRIVE NUMBER',4
ENTMSG    DC.B 'ERROR - REACHED END OF MSDOS DISK.',4

* DATA A EA

          EVEN                        MAKE SURE IT STARTS ON EVEN ADDR
INFCB     DS.B 608                    INPUT FCB

           END FRMSDO
```

```
**************************************************************
* TOMSDOS UTILITY FOR SK*DOS / 68K                           *
* THIS UTILITY TRANSFERS A TEXT FILE FROM SK*DOS FORMAT TO   *
* TO MS-DOS. TRANSLATION TO 6809 CODE IS STRAIGHTFORWARD.    *
**************************************************************

* COPYRIGHT (C) 1986 BY PETER A. STARK

* EQUATES TO SK*DOS

DEFEXT  EQU  $A024      Default extension
FCBCSE  EQU  35         Current sector in buffer
FCBCTR  EQU  34         Current track in buffer
FCBDAT  EQU  96         Beginning of data buffer (256 bytes)
FCBDRV  EQU  3          Logical Drive number
FCBERR  EQU  1          Error code
FCLOSE  EQU  $A008      Close file
FOPENR  EQU  $A005      Open a file for read
FREAD   EQU  $A001      Read the next byte from file
GETCH   EQU  $A029      Get input character with echo (7 bits)
GETNAM  EQU  $A033      Get file name into FCB
GETNXT  EQU  $A020      Get next character from buffer
MAXDRV  EQU   802       Maximum drive number
PERROR  EQU  $A037      Print error code
PSTRNG  EQU  $A035      Print CR/LF and string
SREAD   EQU  $A01C      Read a single sector
SWRITE  EQU  $A01D      Write a single sector
VPOINT  EQU  $A000      Point to SK*DOS variable area
WARMST  EQU  $A01E      Warm start

TOMSDO    BRA.S START                 GO TO START

          DC.W $0100                  VERSION NUMBER
* START OF ACTUAL PROGRAM
START     CLR.B D3                    PREVIOUS CHARACTER WAS NONE
          DC VPOINT                   POINT TO SK*DOS DATA AREA
          MOVE.L A6,A4                POINT TO SYSTEM FCB
          DC GETNAM                   GET FILE SPEC INTO FCB
          BCS.L HELP                  PRINT HELP MESSAGE ON ERROR

* FILE SPEC WAS OK; DEFAULT TO .TXT
          MOVE.B #1,D4                DEFAULT EXTENSION CODE
          DC DEFEXT                   DEFAULT TO .TXT

* NOW GET OUTPUT DRIVE NUMBER
          DC GETNXT                   GET NEXT CHARACTER
          SUB.B #$30,D5               CONVERT FROM ASCII
          CMP.B MAXDRV(A6),D5         CHECK IF VALID
          BHI.L HELP                  IF ERROR
          LEA OUTFCB(PC),A4           POINT TO OUTPUT FCB
          MOVE.B D5,FCBDRV(A4)        STORE DRIVE INTO OUTPUT FCB
          MOVE.W #$0006,FCBCTR(A4)    READY TO READ TRACK 0 SECTOR 6
          DC SREAD                    READ SECTOR FROM MSDOS DISK
          BNE.L ERROR                 68K MUST BE ABLE TO READ
          MOVE.L FCBDAT+4(A4),D7      CHECK BYTES 5-8
          BNE.S ISMSDO                MUST NOT BE ZERO!
NOTMSD    LEA NOTMSG(PC),A4           ZERO MEANS PROBABLY SK*DOS
          DC PSTRNG                   PRINT "NOT A MSDOS DISK - CONTINUE?
          DC GETCH                    GET ANSWER
          AND.B #$DF,D5               CVT TO UPPER CASE
          CMP.B #$59,D5               CHECK FOR Y
          BEQ.S ISMSDO
          DC WARMST                   NO, QUIT

ISMSDO    LEA OUTFCB(PC),A4           POINT TO OUTPUT FCB
          MOVE.W #$0101,FCBCTR(A4)    START AT TRACK 1 SECTOR 1
          LEA OUTFCB+FCBDAT(PC),A1    POINT TO OUTPUT FCB DATA AREA
          LEA 512(A1),A2              AND PAST IT
* NOW ACTUALLY OPEN THE INPUT FILE
          MOVE.L A6,A4                POINT TO USER FCB
          DC FOPENR                   AND GO OPEN THE FILE
          BNE.L ERROR                 IF ERROR

* MAIN LOOP TO READ AND OUTPUT EACH CHARACTER

MAIN      MOVE.L A6,A4                POINT TO USER FCB
          DC FREAD                    GO READ NEXT CHARACTER
          BEQ.S CHAROK                GO ON IF NO ERROR

* IF THERE WAS AN ERROR, SEE IF END OF FILE
          MOVE.B FCBERR(A4),D7        GET ERROR CODE
          CMP.B #8,D7                 COMPARE WITH END OF FILE ERROR
          BNE.L ERROR                 NOT END OF FILE, SO REAL ERROR
          MOVE.B #$1A,D5              ELSE FINIS  WITH CONTROL-Z

* CONTINUE IF CHARACTER IS OK
CHAROK    CMP.B #$0A,D5               IS IT LINE FEED?
          BNE.S OUTPUT                NO, JUST OUTPUT IT
          CMP.B #$0D,D3               WAS PREV CHAR A CR?
          MOVE.B D5,D3                YES, SAVE PREVIOUS CHARACTER
          BRA.S MAIN                  YES, SO SWALLOW IT
OUTPUT    MOVE.B D5,D3                SAVE CHARACTER
          BSR.S PUTFCB                AND PUT INTO OUTPUT FCB
          CMP.B #$1A,D5               END OF FILE?
          BEQ.S QUIT                  YES, JUST QUIT
          CMP.B #$0D,D5               WAS IT RETURN?
          BNE.S MAIN                  NO, SO JUST CONTINUE
```

Continued on page 22



The Macintosh™ Section

Reserved as a

A place for your thoughts

And ours.......

Mac-Watch

SideKick & Accessories

As you are probably aware, if you have been reading 68 Micro Journal for any length of time, we were one of the first magazines to secure and report on the Macintosh™. Actually within weeks after its release. We were excited about it when it arrived, mainly because it was driven by the 68000. However, because of its lack of sufficient RAM for most any practical application, we could not find a whole lot to say about it as a real honest-to-goodness computer. That is until a way was shown to give it the power that Apple left off at first. *Fact is, we were the first to publish complete instructions for upgrading the Mac to 512K. Mike Wolfe, who was instrumental in our scoop on the 64K CoCo also led the way for our 512K Macintosh upgrade.*

Now that the Macintosh Plus has arrived, and Apple improved the System (3.2) and Finder (5.3) applications, we find it a capable system, for most any application but the most demanding. And, within the next few months we should see the arrival of the 68020 Mac, complete with 68881 math co-processor and 2 to 4 Meg of RAM, and that should even satisfy just about all the remaining doubtful.

It now fits nicely into our area of magazine coverage. And even I am surprised at the response we are getting on our Mac-Watch series. What is especially gratifying is that a lot of our new Mac readers are finding out about our excellent OS-9, SK*DOS and other 68XXX systems. So, that should make about everyone happy. I hope that we will soon be *receiving additional input, from our readers, on the Mac. Fact is, we are now looking for someone to author a monthly column on the Macintosh and also someone to do a column on 68XXX assembler programming.* We don't want to displace any of our present coverage, we want to expand. Do I see any hands?

This month we want to tell you about a swell product from BORLAND International - **SideKick**. Actually not *a* product, but more than several products. SideKick & Accessories, The Macintosh Office Manager.

SideKick is a regular *machine Friday,* in that it and its companion accessory - PhoneLink easily attached to any touchtone telephone and will automatically dial any telephone number in the world.

First however, let me tell you about SideKick the application. SideKick is a multiply window program that allows for the automatic dialing (tone/pulse) of your telephone. In addition it does a lot of telephone housekeeping.

1. **SideKick** allows multiple 'phone books' in which you can place any number in the world (direct dialing). It maintains separate data files for several applications.

2. **Phonebook** entries are like an electronic card file. You can enter names, addresses, city, state, zip code, area code, telephone number, company, miscellaneous notations and catego y printer, minister, lawyer, etc.). In addition, it can keep a tally of the time you spent on each call and even compute the charges for each call, as well as time stamping & billing all calls and computing your consulting fees, if that is your purpose. All these functions may be printed out immediately or at a later date, or transported to other documents to suit your particular purpose. You can enter in just a name and number or the whole ball of wax - your choice.

Once entered you just select the name from a menu list and click DIAL. When the phone on the other end answers you click 'OK' and SideKick does the rest. Names selected from the menu are automatically dialed. Numbers dialed also have the information you entered on their record displayed as you make your call.

Another nice feature is that you can enter notes, into a note file and the phone log while on the phone. SideKick knows where to put the new notes as a permanent part of the log.

The phone log may be sorted by several different fields. Sorting can be by name, company or category. Additionally the phone log may be examined on the screen or printed out at a later time. Telephone dialing speed can be regulated to match your local conditions. Full or partial Hayes protocols are supported. And you can even change phone books on the fly, if you maintain more than one.

3. **CalendarBook** is another accessory that resides as a DA in the Apple menu. Events and activities can be entered from SideKick or any other application you might happen to be running. The alendarBook displays a full months calender and a running page or so of any date related notations you might have entered for that date. The current day is highlighted each time the calendar is called.

The monthly calendar displays any month from 1905 to 2030. Days with events are circled on the monthly calendar. You can scan back and forth in the calender to any month you desire, as long as it is between the years above.

Week-at-a-Peek is another menu selection in the calender menu. Here you can enter the notations to flag the days of the month that relate to that particular event. It features the full capability to enter text in a normal style, including word wrap-around. Search features are present in its menu.

4. **AreaCodeLookup** is another DA. It is selected from the Apple menu and displays the locality, region and time-zone of any zip code entered. As they are stored in another DA called Notepad+ (also furnished), you can use the Notepad+ search feature to find the area code of practically any city. This one we use a lot.

5. **Calcula or+** is a real dandy of a DA multidimensional business calculator. It features a 'paper tape' that can be printed out, and is displayed on the screen. Values, functions and operations can be entered from the keyboard or clicking the calculator keys. It supports almost all functions required in any business, including most advanced math functions. This accessory alone is practically worth the price of the entire package!

6. **MacClock** is also a DA. It is a large analog clock with a sweep hand. It gets its settings from the system clock. Nice but really this one didn't grab us too much. However, you don't have to install it. So for those that need a large analog clock, with accurate sweep hand - well, here it is.

7. **MacDialer** is a stripped down version of SideKick (but not by much). It is also a DA and resides in the Apple menu. Now this one we use a lot. It allows automatic dialing without quitting whatever application you might be running. It has the phone notes window, phone book menu, time stamping & billing and phone log features of SideKick. The one thing it lacks is the ability to add or create new phone book entries. Those must be done directly from SideKick. However, its big advantage is that you can run it from within you application.

8. **Notepad+** is another of the DAs in this fine package. It is actually a mini-word processor. It has most of the basic features of some of the more popular Mac editors. its saved files are compa ble with both Word™ and MacWrite™. Notepad+'s windows may be resized or moved, whatever is necessary when its window overlaps your application window.

9. **QuickSheets** is another DA. It is a real neat list and record keeping file in notepad format. There are fours different sheets that comprise the pad. They include Alarms, Expenses, redit ards and Things-to-do. QuickSheets can be printed in pocketbook form by the furnished 'Print Manager' utility. Either blank sheets or sheets with data may be printed. All sheets are saved as text files.

10. **MacTerm** is one of our favorites. This is a Hayes compatible terminal program that runs as a DA. It operates at 300, 1200 or 2400 baud. It features automatic dialing from a directory of commonly used numbers. It is integrated with Notepad+ for sending and receiving data. MacTerm supports the normal EDIT menu and accepts automatic dialing or the regular Hayes type 'AT' keyboard dialing.

As you probably know we port many of our articles from a FLEX and OS-9 GIMIX III 6809 computer (those we receive on disk). We have a cable from the Mac to the CRT port of the GIMIX and download in that manner.

Up until we received this package we were having problems with our normal Macintosh terminal program in our porting process. The received file was getting some nulls inserted into the text and some of our other applications complained about such behavior. In fact the publishers of the spelling checker we are using (a review coming soon) even went so far as to write a special Macintosh routine to filter out those troublesome nulls just for our use. That is what I call *Super Superior support*

(SpellsWell is the name of the checker), and it is the best we have ever used!

However, we changed over to MacTerm and it has behaved as a proper terminal program should. In addition it can be running while we have another application active. That is a great time saver and necessary when in the process of communicating we need to refer to some file on the system.

11. **RediPrinter** is another very useful DA, perhaps, for some worth the price alone.

RediPrinter is a print spooler (not LaserWriter) that prints your file while you go on about your Macing business. For those longer printing jobs, this is a real winner.

SideKick and all its accessories are **not** copy protected, which means you can install them on your hard disk or not have to go through all that stupid hassle that some other vendors ask us to endure.

All in all, we find this package well worth the price. Fact is, as I stated, several of the accessory programs are well worth the price alone. All together it is a bargain. And what is even better, it all works as the book says. Which is not the case with a lot of other stuff we have reviewed recently.

Speaking of the book, well, there are many other nice features of those related above that I just don't have the space to outline. The book is jam packed with over a 150 pages of how-to stuff. All written and accompanied with pictures for even those of us who have problems chewing gum and walking all at the same time. The documentation and update sheets *leave nothing to be discovered, it is accurate and complete.*

We recommend this product highly. SideKick can be ordered direct from **BORLAND International** or through many Mac software retailers.

BORLAND International
4585 Scotts Valley Drive
Scotts Valley, CA 95066

Price: $99.95
Including the PhoneLink hookup accessory

If you happen to decide to add this to your Macintosh - tell them we sent you. O.K.?

EOF

A Staff Review

READING AND WRITING MS-DOS AND PC-DOS DISKS UNDER SK*DOS

Continued from page 19

```
         MOVE.B #$0A,D5
         BSR.S PUTFCB             FOLLOW WITH LF
         BRA.S MAIN               AND ALSO CONTINUE

* PUTFCB - PUT CHARACTER INTO OUTPUT FCB, AND WRITE IT OUT IF FULL

PUTFCB   MOVE.B D5,(A1)+          PUT INTO FCB
         CMP.B #$1A,D5            END OF FILE?
         BEQ.S WRITE              YES, GO WRITE IT
         CMP.L A1,A2              PAST END?
         BNE.S RTS                NO, EXIT
WRITE    MOVE.B D5,D2             TEMP SAVE CHARACTER
         LEA OUTFCB(PC),A4        POINT TO OUTPUT FCB
         DC SWRITE                WRITE IT OUT
         BNE.S ERROR              IF ERROR
         MOVE.L A6,A4             POINT BACK TO USER FCB
         LEA OUTFCB(PC),A0        POINT TO OUTPUT FCB
         LEA FCBDAT(A0),A1        POINT TO OUTPUT FCB DATA AREA
         ADD.B #1,FCBCSE(A0)      GO TO NEXT SECTOR
         MOVE.W FCBCTR(A0),D7     CURRENT TRACK AND SECTOR
         CMP.B #10,D7             PAST SECTOR 9?
         BNE.S ALMRTS             NO, OK TO EXIT
         ADD.W #$0100,D7          YES, SO NEXT TRACK
         MOVE.B #1,D7             AND SECTOR 1
         MOVE.W D7,FCBCTR(A0)     PUT BACK INTO FCB
         CMP.W #$2801,D7          PAST TRACK $27?
         BNE.S ALMRTS             NO, OK TO CONTINUE
         LEA FULMSG(PC),A4
         DC PSTRNG                PRINT "DISK IS FULL
         BRA.S QUIT               AND QUIT
ALMRTS   MOVE.B D2,D5             RESTORE CHARACTER INTO A
RTS      RTS                      AND EXIT

* CLOSE SUBROUTINE

CLOSE    MOVE.L A6,A4             POINT TO FCB
         DC FCLOSE                GO CLOSE THE FILE
         RTS                      AND THEN RETURN W/O ERROR MESSAGE
* HELP ROUTINE

HELP     LEA HLPMSG(PC),A4
         DC PSTRNG                PRINT HELP MESSAGE
         DC WARMST

* ERROR ROUTINE

ERROR    DC PERROR                PRINT ERROR CODE
QUIT     BSR.S CLOSE              CLOSE THE FILE
         DC WARMST                AND RETURN TO SK*DOS

* TEXT STRINGS

HLPMSG   DC.B 'TOMSDOS writes a text file to a specially-formatted'
         DC.B $0D,$0A
         DC.B 'MSDOS disk. The correct syntax is'
         DC.B $0D,$0A
         DC.B '  TOMSDOS <SK*DOS file spec> <MSDOS drive number>',4
NOTMSG   DC.B 'OUTPUT DISK IS PROBABLY NOT A MSDOS DISK - CONTINUE? ',4
FULMSG   DC.B 'MSDOS DISK IS FULL.',4

* DATA AREA

         EVEN                     MAKE SURE TO BE ON EVEN ADDRESS
OUTFCB   DS.B 576

         END TOMSDO

------------------------------------------------------------------

+++
```

Reversi

By: Werner F.W. Zychlinski
Peaclstr. 22
2820 Bremen 70
W. Germany

To all OS/9 users: here is a strong game for enjoy and fun. It is the game of "Reversi", also named "Othello". The original version is programmed by Ed. Wright in the FORTRAN language. This original program runs here on High-school on a HARRIS /4.

I have "hand-compiled" the original FORTRAN program into pure 6809 assembler code. The result runs under FLEX9. For cursor addressing it requires the terminal characteristics. This can be made with the "CRTSET" utility from FHL or manually.

The following version of the Reversi game was programmed and tested under OS-9/68000 V1.2 with BASIC09 V1.2. It should be run on each OS-9/68000 system, because it is hardware independent. The program requires at least 33k (68000) of RAM storage. Because BASIC09 for the 68000 and the 6809 would be the same, this program can also be run under the OS-9/6809 and BASIC09, but I can't test it.

Any one who haven't time and mind to type in the listing but like to have the program can send me $15. I can send the program on disk in the following two formats:

5" 40 Trk DS DD   5" 80 Trk DS DD

I can format the disk only on the OS-9/68000 system, but I think it can be also read on a OS/9-6809 system. Microware notes, that the two disk formats are compatible, but I can't test it.

PS: The FFT program with the Uhrich algorithm uses 40sec on a 68000 with 8 Mhz, software floating-point and BASIC09.

```
 PROCEDURE moveg
  PARAM b(10,10),oc,nm,movesi(30),movesj(30):INTEGER
  PARAM dirr(30,8),lc(30),jaa(8),iaa(8),im,nomove,nflip(30):INTEGER
  DIM i,j,l,ia,ja,iv,mvi,mvj,lcc,ld:INTEGER
 DIM icc:BOOLEAN
 DIM char:STRING[1]
 FOR i=1 TO 30
lc(i)=0
 nflip(i)=0
NEXT i
im=0
 FOR i=2 TO 9
 FOR j=2 TO 9
  IF b(i,j)<>0 THEN 20
icc=FALSE
 FOR l=1 TO 8
 ia=iaa(l)
 ja=jaa(l)
  IF b(i+ia,j+ja)-oc<>0 THEN 5
iv=1
 4 iv=iv+1
 mvi=i+iv*ia
 mvj=j+iv*ja
  IF b(mvi,mvj)=0 THEN 5
  IF b(mvi,mvj)=100 THEN 5
  IF b(mvi,mvj)=oc THEN 4
 IF NOT(icc) THEN
 im=im+1
 icc=TRUE
ENDIF
  nflip(im)=nflip(im)+iv
  lc(im)=lc(im)+1
 ld=lc(im)
  dirr(im,ld)=l
 5 NEXT l
 IF icc THEN
  movesi(im)=i
  movesj(im)=j
ENDIF
 20 NEXT j
NEXT i
  IF im>0 THEN 30
 IF nm=oc THEN
  PRINT "I don't see a move for me!"
ENDIF
 30 END
  PROCEDURE count
   PARAM b(10,10),oc,noc:INTEGER
  DIM i,j:INTEGER
 noc=0
  FOR i=2 TO 9
  FOR j=2 TO 9
  IF b(i,j)=oc THEN
 noc=noc+1
ENDIF
NEXT j
NEXT i
END
  PROCEDURE boardp
   PARAM board(10,10),nm,nhd:INTEGER
   DIM nmp,i,j,io,ix:INTEGER
  io=0\ix=0
  nmp=nm-nhd
  FOR i=2 TO 9
  FOR j=2 TO 9
  IF board(i,j)=-1 THEN
 io=io+1
ENDIF
  IF board(i,j)=1 THEN
 ix=ix+1
ENDIF
NEXT j
```

```
 NEXT i
PRINT
   PRINT " j= 1  2  3  4  5  6  7  8"
   PRINT "i "; TAB(40); "after "; nmp; " moves:"
  FOR i=2 TO 9
 PRINT i-1;
  FOR j=2 TO 9
  IF board(i,j)=-1 THEN
  PRINT "  O";
ENDIF
  IF board(i,j)=0 THEN
  PRINT "  .";
ENDIF
  IF board(i,j)=1 THEN
  PRINT "  X";
ENDIF
 NEXT j
 IF i=3 THEN
   PRINT TAB(40); "'X' occupies "; ix; " pieces";
ELSE
 IF i=6 THEN
·  PRINT TAB(40); "'O' occupies "; io; " pieces";
ENDIF
ENDIF
PRINT
 NEXT i
END
  PROCEDURE boardc
   PARAM movesi(30),movesj(30),iff,iaa(8),jaa(8):INTEGER
   PARAM b(10,10),oc,dirr(30,8),lc(30):INTEGER
   DIM mi,mj,i,l,ia,ja,iv,mvi,mvj,ndir:INTEGER
  mi=movesi(iff)
  mj=movesj(iff)
  b(mi,mj)=-(oc)
  ndir=lc(iff)
  FOR i=1 TO ndir
  l=dirr(iff,i)
  ia=iaa(l)
  ja=jaa(l)
 iv=0
LOOP
 iv=iv+1
  mvi=mi+iv*ia
  mvj=mj+iv*ja
  EXITIF b(mvi,mvj)=-(oc) THEN
 ENDEXIT
  b(mvi,mvj)=-(oc)
 ENDLOOP
 NEXT i
END
  PROCEDURE moves
   PARAM b(10,10),oc,nm,movesi(30),movesj(30),nflip(30):INTEGER
   PARAM dirr(30,8),lc(30),im,iff,iaa(8),jaa(8):INTEGER
   DIM bt(10,10),btt(10,10),dirb(30,8):INTEGER
   DIM btts(9,9,20),dirbb(30,8):INTEGER
   DIM mbi(30),mbj(30),lcb(30),nflipb(30),iy(24),jy(24):INTEGER
   DIM imid(24),jmid(24),id(24),jd(24),nocmi(4),nocmj(4):INTEGER
   DIM mbbi(30),mbbj(30),lcbb(30),nflib(30):INTEGER
   DIM i,j,k,l,il,jl,k1,kj2:INTEGER
   DIM ico,mi,mj,nd,ia,ja,iv,mvi,mvj,naibo,ic,ll,ipp,jpp:INTEGER
   DIM im1,im2,iz,jz,mk,ml,ill,jll,iq,jq,kc1,kc2:INTEGER
  DIM oflipm,noc,nomve:INTEGER
  DATA 2,2,9,9,2,9,9,2
   DATA 3,4,5,6,7,8,9,9,9,9,9,9,8,7,6,5,4,3
   DATA 2,2,2,2,2,2,2,2,2,2,2,2,3,4,5,6,7,8
   DATA 9,9,9,9,9,9,8,7,6,5,4,3
   DATA 5,1,3,8,1,6,9,1,9,9,1,9,6,1,8,3,1,5,2,1,2,2,1,2
   DATA 2,1,2,2,1,2,5,1,3,8,1,6,9,1,9,9,1,9,6,1,8,3,1,5
   DATA 4,1,4,7,1,7,9,1,9,9,1,9,7,1,7,4,1,4,2,1,2,2
   DATA 1,2,2,1,2,2,1,2,4,1,4,7,1,7,9,1,9,9,1,9,7,1,7,4,1,4
  FOR i=1 TO 4\READ nocmi(i)\NEXT i
   FOR i=1 TO 4\READ nocmj(i)\NEXT i
   FOR i=1 TO 24\READ id(i)\NEXT i
   FOR i=1 TO 24\READ jd(i)\NEXT i
   FOR i=1 TO 24\READ iy(i)\NEXT i
   FOR i=1 TO 24\READ jy(i)\NEXT i
   FOR i=1 TO 24\READ imid(i)\NEXT i
   FOR i=1 TO 24\READ jmid(i)\NEXT i
 ico=0
 iff=1
  IF nm=59 THEN 20
  FOR i=1 TO im
  mi=movesi(i)
  mj=movesj(i)
  IF mi<>3 AND mi<>8 THEN 13
  IF mj<>3 AND mj<>8 THEN 13
  IF mi=3 AND mj=3 THEN
ic=1
ENDIF
  IF mi=3 AND mj=8 THEN
ic=2
ENDIF
  IF mi=8 AND mj=8 THEN
ic=3
ENDIF
  IF mi=8 AND mj=3 THEN
ic=4
ENDIF
   IF b(nocmi(ic),nocmj(ic))=0 THEN
  nflip(i)=nflip(i)-50
ENDIF
  13 IF mi<>2 AND mi<>9 THEN 11
  IF mj<>2 AND mj<>9 THEN 11
 ico=ico+1
  nflip(i)=nflip(i)-60
  11 IF mi<=3 OR mi>=8 THEN 2
  IF mj<=3 OR mj>=8 THEN 2
  nflip(i)=nflip(i)+10
GOTO 12
 2 nd=lc(i)
 FOR j=1 TO nd
 l=dirr(i,j)
 ia=iaa(l)
 ja=jaa(l)
iv=1
 4 iv=iv+1
  mvi=mi+iv*ia
  mvj=mj+iv*ja
  IF b(mvi,mvj)-oc=0 THEN 4
LOOP
 iv=iv+1
  mvi=mi+iv*ia
  mvj=mj+iv*ja
  EXITIF b(mvi,mvj)-oc=0 THEN
ENDEXIT
  IF b(mvi,mvj)+oc<>0 THEN 5
ENDLOOP
  8 IF b(mi-ia,mj-ja)<>0 THEN 5
  9 nflip(i)=nflip(i)-5
GOTO 12
5 NEXT j
 12 NEXT i
 FOR i=1 TO im
naibo=0
 mi=movesi(i)
 mj=movesj(i)
ic=0
LET bt=b
l=0
 FOR j=1 TO 24
 ipp=id(j)
 jpp=jd(j)
```

```
  IF movesi(i)<>ipp OR movesj(i)<>jpp THEN 56
k=j
 56NEXT j
  RUN boardc(movesi,movesj,i,iaa,jaa,bt,oc,dirr,lc)
  RUN moveg(bt,-(oc),nm,mbi,mbj,dirb,lcb,jaa,iaa,im1,nomve,nflipb)
 IF im1<>0 THEN 63
 nflip(i)=nflip(i)+100
GOTO 32
 63 FOR j=1 TO im1
LET btt=bt
  RUN boardc(mbi,mbj,j,iaa,jaa,btt,-(oc),dirb,lcb)
 IF ll=0 THEN 38
ic=1
 iz=iy(ll)
 jz=jy(ll)
  IF b(iz,jz)+oc=0 THEN
 mk=jmid(ll)
 ml=imid(ll)
  IF b(ml,mk)=0 THEN
 nsubo=90
ENDIF
ENDIF
  IF btt(mi,mj)=oc THEN
  nflip(i)=nflip(i)-40
ic=2
ENDIF
  38RUN count(btt,-(oc),noc)
 IF noc<=0 THEN
  nflip(i)=nflip(i)-200
 GOTO 32
ENDIF
  42 FOR k1=2 TO 9
  FOR k2=2 TO 9
   btts(k1,k2,j)=btt(k1,k2)
 NEXT k2
 NEXT k1
  FOR il=2 TO 9
  FOR jl=2 TO 9
   IF btt(il,jl)=0 THEN 100
   96 IF btt(il,jl)-oc=0 THEN 100
  99 FOR iz=1 TO 8
iv=0
  80iv=iv+1
   ill=il+iv*iaa(iz)
   jll=jl+iv*jaa(iz)
   IF btt(ill,jll)=0 THEN 36
   IF btt(ill,jll)-100=0 THEN 36
   IF btt(ill,jll)-oc<>0 THEN 80
  90NEXT iz
  100NEXT jl
 NEXT il
   95RUN moveg(btt,oc,nm,mbbi,mbbj,dirbb,lcbb,jaa,iaa,im2,nomve,nflib)
  IF im2=0 THEN 103
  FOR il=1 TO im2
   IF mbbi(il)<>2 OR mbbi(il)<>9 THEN 102
   IF mbbj(il)<>2 OR mbbj(il)<>9 THEN 102
 GOTO 36
  102NEXT il
   103nflip(i)=nflip(i)-190
 36NEXT j
  IF ic<>1 THEN 35
  FOR k=1 TO 24
 iq=id(k)
 jq=jd(k)
  IF mi=iq AND mj=jq THEN 50
  IF b(iq,jq)+oc<>0 THEN 50
  FOR k1=1 TO im1
  IF btts(iq,jq,k1)=oc THEN
  nflip(i)=nflip(i)-8
ENDIF
 NEXT k1
 50NEXT k
  nflip(i)=nflip(i)+25-nsubo
  35 FOR k=1 TO 4
 kc1=ncorni(k)
 kc2=ncornj(k)
  IF b(kc1,kc2)<>0 THEN 60
  FOR k1=1 TO im1
  IF btts(kc1,kc2,k1)=oc THEN
 nflip(i)=nflip(i)-55
ENDIF
NEXT k1
  IF ic<=1 THEN 60
   IF mi=kc1 AND mj=kc2 THEN 60
  FOR k1=1 TO im1
   IF btts(kc1,kc2,k1)=oc THEN
  nflip(i)=nflip(i)-20
ENDIF
 NEXT k1
 60NEXT k
 32NEXT i
  nflipm=-800
  FOR i=1 TO im
   IF nflip(i)>=nflipm THEN
  nflipm=nflip(i)
 ifs=i
ENDIF
NEXT i
20END
  PROCEDURE handic
   PARAM oc,b(10,10),nhd:INTEGER
  DIM nah:INTEGER
  DIM respon:STRING[2]
nhd=0
   INPUT " Do You like a handicap (y/n)? ",respon
   IF respon="y" OR respon="Y" THEN
 nah=-(oc)
REPEAT
   INPUT " How many pieces (1-4)? ",nhd
  UNTIL nhd>0 AND nhd<5
   RUN handi(b,nhd,nah,oc)
ELSE
    INPUT " Do You give me an handicap (y/n)? ",respon
   IF respon="y" OR respon="Y" THEN
 nah=oc
REPEAT
   INPUT " How many pieces (1-4)? ",nhd
  UNTIL nhd>0 AND nhd<5
   RUN handi(b,nhd,nah,oc)
ENDIF
ENDIF
END
 PROCEDURE handi
   PARAM b(10,10),nhd,nah,oc:INTEGER
   DIM ncorni(4),ncornj(4):INTEGER
   DIM i,j,i1,i2,sign:INTEGER
   DATA 2,2,9,9,2,9,9,2
   FOR i=1 TO 4\READ ncorni(i)\NEXT i
   FOR i=1 TO 4\READ ncornj(i)\NEXT i
 sign=-1
  IF nah=oc THEN
 sign=1
ENDIF
  FOR i=1 TO nhd
  i1=ncorni(i)
  i2=ncornj(i)
   b(i1,i2)=sign*oc
 NEXT i
END
```

```
PROCEDURE reversi
 (* ****************************************************
 (* *                                            *
 (* *            R E V E R S I                       *
 (* *                                            *
 (* * Reversi (Othello) game in BASIC09 under the     *
 (* * OS-9/68000 operating system                  *
 (* * Modelled after an FORTRAN program from          *
 (* * Ed. Wright                                 *
 (* *                                            *
 (* *          Procedures:  boardc                  *
 (* *                       boardp              *
 (* *                       count               *
 (* *                       handi               *
 (* *                       handic              *
 (* *                       movec               *
 (* *                       moveg               *
 (* *                   and reversi             *
 (* *                                            *
 (* * Version 1.01 from 16.Oct.86  Zych              *
 (* *                                            *
 (* *                                            *
 (* *     Werner F.W. Zychlinski                        *
 (* *     Pezelstr. 22                            *
 (* *     2820 Bremen 70                             *
 (* *     W. Germany                             *
 (* ****************************************************

  DIM b(10,10),dirr(30,8),iaa(8),jaa(8):INTEGER
  DIM movesi(30),movesj(30),lc(30),nflip(30):INTEGER
  DIM oc,i,j,nm,nhd,im,movei,movej,nomve,iff,noc,nc:INTEGER
 DIM respon:STRING[2]
  DATA -1,-1,-1,0,1,1,1,0
  DATA -1,0,1,1,1,0,-1,-1
  FOR i=1 TO 8\READ iaa(i)\NEXT i
  FOR i=1 TO 8\READ jaa(i)\NEXT i
  22 FOR i=1 TO 10
 FOR j=1 TO 10
 b(i,j)=0
  IF i=1 OR i=10 THEN
 b(i,j)=100
ENDIF
  IF j=1 OR j=10 THEN
 b(i,j)=100
ENDIF
NEXT j
 NEXT i
 b(5,5)=1
 b(5,6)=-1
 b(6,5)=-1
 b(6,6)=1
 PRINT\PRINT\PRINT
  PRINT " ==================================================="
  PRINT " Welcome to the game of REVERSI on BASIC09"
  PRINT " ==================================================="
PRINT
  INPUT " Do You like to start the game with the 'X' pieces (y/n)? ",respon
oc=1
  IF respon<>"y" AND respon<>"Y" THEN 11
 RUN handic(oc,b,nhd)
nm=nhd
 PRINT \ PRINT \ PRINT \ PRINT \ PRINT \ PRINT
 RUN boardp(b,nm,nhd)
 8 IF nm=60 THEN 15
 R UN moveg(b,-(oc),nm,movesi,movesj,dirr,lc,jaa,iaa,im,nomve,nflip)
 IF im=0 THEN 12
  14 INPUT " Please enter You move (i,j):",movei,movej
movej=movej+1
movei=movei+1
FOR i=1 TO im
 IF movesi(i)=movei AND movesj(i)=movej THEN 13
NEXT i
 PRINT " Error in You move, please enter again!"
 RUN boardp(b,nm,nhd)
GOTO 14
13 nm=nm+1
 RUN boardc(movesi,movesj,i,iaa,jaa,b,-(oc),dirr,lc)
 RUN boardp(b,nm,nhd)
GOTO 2
11 oc=-1
 RUN handic(oc,b,nhd)
b(5,7)=1
b(5,6)=1
nm=nhd+1
 RUN boardp(b,nm,nhd)
GOTO 8
  12 PRINT " I don't see a move for You,";
  PRINT " i am searching for a move for me now..."
 2 IF nm=60 THEN 15
  R UN moveg(b,oc,nm,movesi,movesj,dirr,lc,jaa,iaa,im,nomve,nflip)
 IF im=0 THEN 20
  RUN movec(b,oc,nm,movesi,movesj,nflip,dirr,lc,im,iff,iaa,jaa)
 movei=movesi(iff)-1
 movej=movesj(iff)-1
  PRINT "My move: "; movei; ","; movej
  R UN boardc(movesi,movesj,iff,iaa,jaa,b,oc,dirr,lc)
nm=nm+1
 RUN boardp(b,nm,nhd)
GOTO 8
  20 INPUT "Do You have a move (y/n)? ",respon
 RUN boardp(b,nm,nhd)
  IF respon<>"y" AND respon<>"Y" THEN 88
GOTO 8
 88 IF im<>0 THEN 2
  15RUN count(b,oc,noc)
 RUN count(b,-(oc),nc)
 IF noc>nc THEN
  PRINT "***** Congratulations, You have won !!! *****"
ENDIF
 IF noc<nc THEN
 PRINT "I have won !"
ENDIF
 IF noc=nc THEN
 PRINT "Undecided !"
ENDIF
  INPUT "Do You like to play again (y/n)? ",respon
  IF respon="y" OR respon="Y" THEN 22
END
```

EOF

Ramblings:

Some old, some new, some never come true. But, beware, many do!

rumors and such

DMW

Editor's Note:

Recently I received a well edited and put together CoCo users newsletter. Chock full of nice things about the CoCo, and the hopes of the editor for the future of the CoCo. I receive newsletters and club bulletins from many clubs and other CoCo support and user organizations. I appreciate their loyalty to the CoCo, it has been a gateway to many who would, otherwise, not have had an opportunity to get into computing as a hobby, or for some, pretty serious computer stuff. There was a time when the price/performance of the CoCo could not be beat.

When the CoCo arrived on the scene, it was a great buy, even considering the initial price (nearly $600.00) and small amount of installed RAM. Today it is less than a hundred bucks but not nearly so much a bargain (CoCo II). Newer and more powerful systems have invaded the low-end territory (Atari, Amiga, Big Blue clones, etc.), and that is going to make a difference. Now, the challenge is passed to the recently arrived, but long promised CoCo III. It will have to carry the banner. That may be a stump not easily pulled!

However, let me get back to the newsletter. The editor, as I read his entire issue (only received this one), seems to be a nice sort of fellow. He certainly appears to understand the CoCo, and his loyalty to the CoCo is readily apparent. For that I admire him, we are sorta brothers therein, we both depend on the loyalty of like fellows (and gals) to keep things rolling. *Loyalty is certainly one of those 'short supply' items in today's society, but not here. I know, if it weren't for the thousands of loyal contributors and readers of 68 Micro Journal, you and I would not have this. It is loyalty and the willingness to share, that is overbearingly important. To me, loyalty has an uplifting quality surpassed by few other traits. I will always be grateful for the loyalty, kindness, and many times inspiration showered on me by our contributors and readers, and those faithful advertisers that have put their trust in us. It has been more like a family than a business.* So, I know where my editor friend is coming from.

I will not try to quote directly, I don't desire to engage either him, or any of his loyal readers into a state of discord. I hope he and all his readers, and the newsletter, all the very best, and may their CoCo never drop a bit! However, I will reply to some of his remarks that were directed towards me personally, and 68 Micro Journal.

It should also be realized that thousands of CoCo users are and have been readers of this publication. I have a vested interest in the CoCo, it's wide-spread distribution and well-being. But, *I am not burdened with rose colored glasses, or obligations to anyone but you, the reader. And one thing that needs to be kept firmly in mind, during this discussion-the CoCo is not a religion!*

The thrust of his remarks (made kindly but pointedly) editorially was that I had, in his opinion (apparently), become something of an enemy to the CoCo. Not so, I have one at home, set up right beside my bed...honest injun. I really do, I use it to play chess. It is a great form of relaxation and I can save the game moves for later review, if I desire. For the price and the quality of the graphics, it was, and still is well worth the cost to me. The CoCo II and the chess game (both bought retail) were worth the price, for that purpose. However, I do not have one in my office, it is not sufficient for that purpose considering other alternatives. Although over at the S.E. MEDIA Division and the Data-Comp Division, there are several, with all kinds of disks, modems, RGB monitors, CRT terminals, memory expansions, extension slots and literally thousands of programs, mostly games, on disk and tape as well as cartridge. They are running OS-9, FLEX, SK*DOS, RS DOS and several other op systems that never saw the light of commercial day. However, the focus of applications each does is quite narrow, due to limitations that should not have been.

For instance we do disk duplicating on a CoCo for most all our 5 inch floppy software sales. We have software that can duplicate any media (5 inch) format. S.E. MEDIA and Data-Comp have used them for years. We bought one of the first 200 that was originally shipped. And we were one of the first to sell the CoCo into any public school system, years ago. We spent a lot of time and effort supporting the CoCo, not to mention money. *Also 68 MICRO JOURNAL was the very first computer magazine to do anything of substance on the CoCo. Period!*

Our CoCo column, which ran for years, authored by Bob Nay, was the first ever regular, and longest running CoCo column. From our facilities came the first floppy for the CoCo, first 64K conversion, first 32 plus character screen drivers, first external video terminal drivers, first porting of another operating system to the CoCo, and much, much more. We tried!

We supported it when it made some of our advertisers unhappy (and we certainly needed them, but I hoped it would expand the 68XX user base, that being good for all of us). I still felt, at that time, that the CoCo had a great future. After all it had the 6809 CPU and I was privy to the fact that OS-9 would be there some day, full bore, and with the addition of OS-9, officially by Tandy, the initial shortcomings would be addressed, for the better. FLEX (FHL and Data-Comp versions were already running on the CoCo) and I was hearing from inside Tandy (yes, I do have a few friends around) that they (Tandy) were going all out to promote it as a viable alternative to the Apple and several other lesser known systems, especially in the educational field. Also, of course, we would get lots of new subscribers (that alone would have been reason enough). Alas, it never materialized. They left it lame and lacking, then and now! Seems they were always redirecting the theme of what it was and what it was really supposed to do (education one month, business(?) the next, and ......). Until finally it seemed that all it was going to be allowed to do was games, and play stuff, or a halfhearted attempt to be something it could not be...a real computer. The only reason it sold as many as it did was because of price not versatility, nor quality (per reader survey). Price, for what you get, has always been it's strong suite. And for many that was sufficient.

Now, there is certainly nothing wrong with that, as a number of CoCo users can attest. *But, it is not now, as in the past, been all that it could be for essentially the same cost of manufacture.* And that is, and always has been, the thrust of my complaints. I accept it for what it is, and lament for what it could have been.

The best thing that ever happened to the CoCo was OS-9. But bit banging I/O, a playtoy keyboard (never did they ever find out about some things like control, escape, etc. keys), no expansion slot(s), (an expansion box yes, but at additional cost and not the solution), a video mapped character set that strained beyond about 48 characters a line, or so, and the beat goes on.

Sounds like I got a burr for the CoCo? Well, if you nodded yes, your dead wrong! I've got the burr for Tandy. They birthed it sick, never made it well and never even tried, they just let it lay out there and wimper. It has been like seeing someone you care for stricken, and who was or could have been fine and physically strong, but was neutered at birth! The CoCo always, well, almost, had all the basics to be a world-beater. But just enough essential stuff was left out so that it could not quite make it as a serious contender. I think I know why, but then that is another story.

If you are completely satisfied with your CoCo, as it comes out of the box, then it really doesn't matter what I say. Especially if you are one of those who bought it just for fun and games, like I did. If so, then you got your moneys worth. I know I did. But as a serious computer..never! Our files are full of letters from readers who kept asking, what can I do to make my CoCo a serious system? Years ago we did all we could do, and in my opinion a lot more than Tandy had done up to that time (building support, adapting additional support products, etc.) but up-stream is a hard pull, and we bad other, more responsive duties to attend. And that was that!

The nice editor said that I was taking swipes at the CoCo unfairly, according to a remark I made here, some months back, along lines such as above. He went on to point out some dire happenings to others who had done likewise. I sincerely hope they never happen to either of us, we need each other too badly, especially with the looks of things to come. We only survive because we stick together. Even if we disagree, it should be done with the realization that nothing is carved in stone, except death and taxes. And neither of us can do much about either one. Except perhaps hasten their coming.

If be is all that upset about the state of the CoCo scene, then I suggest that he and all who feel as he does, should take it directly to the ones really responsible. For years I have. Certainly we all know who that is.

However, I think I am beginning to see the end of the new CoCo III, even before it gets out of diapers. It should be given a chance. I sincerely hope I'm wrong!

First, there is a matter of expected longevity and support. No one wants to invest in a product that is on the way out. For when it is gone, also gone is the support, making it a sad and bad investment. The government may let you write off a computer system in a few short years, but it's manufacturer should be a little more respectiful of the sweat we have to put out to buy his product! Remember the Model One? Try to order parts for one. Regardless of what they might tell you, you will still be out of luck for many parts. At least that is the story at my local Tandy outlet. But if you had bought a SWTPC, GIMIX, etc., you can still get parts. Now, is the CoCo III on the way out? Maybe. Fact is, I think the delay of it's arrival has a lot to say, about that. Also the manner in which it is being advertised by Tandy seems to be the path that all the other defrocked systems took in days past. *Remember the Model MC-10 Micro Color Computer, Tandy catalog number 26-3011?* How much support should those users have expected? Even if some parts and other items are still available for these forgotten machines, I believe the vendor has an obligation to keep the thing in stock channels long enough to build a user base that will support the system. Otherwise if it is of such quality that, from it's first day it has little chance of real acceptance, and it is still tossed out to the market knowing it is a cripple to start, then we have been had! It all boils down to a simple principal - if it isn't worthy of your reputation don't sell it! Also, if you are not going to give it wholehearted support, I want to know! I don't want another MC-10.

Now, I don't know how long it will take, but I feel that a system that is discounted within a few months of it's introduction, should be looking over it's shoulder. Considering that most all the other *dropped* systems went along the same path. Some things just naturally have the smell of 'by-by'.

Recently I received Radio Shack flyer, number 409. On page 14 is a full page covering the CoCo III, and already the discounting starts. I hope that this is not the *final days trend* (as has befallen other systems). Also, not one mention of OS-9. *The one thing that makes the CoCo stand out...OS-9!* Have you ever considered how a CoCo running OS-9 compares with their other high-end stuff. And the FLEX versions, both FHL and Data-Comp made it a real hummer. Stylo against their word processor for the CoCo. DynaCalc against that other spreadsheet(?), the beat just keeps going. Especially back a year or so. Sorta like David and that big guy!

Even with all the built in (or should I say left out) features, given a good video monitor and keyboard (available from vendor sources) and some I/O expansion add--on resources, the CoCo

III could be a winner. If the pricing is made right. However, by the time all the necessary extra hardware is rounded up (and paid for), including dual disk drives, decent software and the RAM expansion to 512K (end of line), the price is not too attractive. Especially when you consider what is available today (heavily discounted in many stores and computer shops) with a faster, more powerful CPU (68XXX), better graphics, built in disk (one, gotta buy just one here, and most places less expensive), far superior keyboard, more RAM standard, quality monitor and able to run a better version of OS-9 (68K,which is terrific), and much less costly as a package than the CoCo III after you add on all the necessary stuff, well, it makes you kinda wonder what the future holds.

Now, to my editor friend, I ask, what do you think? Who decreed that it should be a wimp? Who decided that it should not have a real serial port like all the others (cost less than a buck or so to put one on during manufacture), *not even just one?* Who hung in there with a keyboard that isn't even a pretender? Despite there being a loud cry from loyal CoCo users for many a year. I know they must have heard that long and loud. Who spent an entire page of national (and maybe beyond) advertising and not one word about the best part of the entire system...OS-9? Sure they have 24X80 now, but we gave it to CoCo users years ago. Ever try to look at 80 character lines on your TV? Fun, huh? Sure if you buy all the other stuff to get it sorta up to speed, the price just goes through the ceiling. What sort of good deal is that? Who decided all that, and more? ***Well, my friend, not me!***

For if you have been reading my stuff these years you would know that I have long yelled (editorially) about those short comings, and more. I wanted it to make it, I spent a lot of money on a CoCo magazine that was doing o.k. but too technical for most users (Color Micro Journal). Fact is, when I decided to pull it, it was because of two things - no support for the serious user, except for a few, who did advertise and support, and a lot more who never got serious. Fact is, when it ceased publication it was the second largest CoCo publication around, and there were more than several then. I can count on my fingers those that tried to make it a *real* computer (products, articles, etc.). Bob Nay, Steve Odneal, Dale Puckett, FHL Labs, MicroWorks, Peter Dibble, to name a few right off the top of my head (and of course, us). We all put a lot of serious user effort into the thing, so if Tandy insist it being born weak, then it is like the upstream boat with no get-up-and-go, and please don't get mad at me about it!

You see, my friend, I am not the enemy. I paid my dues. You might not have been aware but we were ardent supporters when it first came out. *Even Tandy, in their official publications stated that if you wanted to get to the heart of the CoCo, then 68 Micro Journal was the magazine to read.* And there were several other CoCo magazines at the time also. But as time passed, and it seemed to be that it was foretold that the CoCo was always to be a semi-toy, then, and only then I began to holler. About all it got us was no support from Tandy, and quite a bit for those other magazines that settled for less than was possible.

However, I would go out today and buy a CoCo if I didn't have access to other better or less expensive OS-9 or games (or both) systems. You see, I feel that OS-9 Level II gives it the ----s that Tandy left off. Even with the mickey mouse keyboard, bit banger serial port, and all the other shortcomings I mentioned, and a lot I didn't. If the price is shaved to the level of the others, and it's base support is ready for serious application, then there may be hope. Unless, of course, it just disappears first. Anyway it turns, I ain't gonna get rid of mine. It plays chess pretty good.

Again, sorry I poked you wrong, didn't mean to. Seems I am always getting into it because I dare try to tell it as I see it. But it doesn't look good to me. But, if I am wrong (hope I am), then we all will be better off. Right? The more CoCo users, the more 6809 users, and that is a valuable asset to us here at 68 Micro Journal! Have a good 'un.

DMW

☎ (615)842-4600

Telex 6106006630

South East Media

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

SPECIAL

K-BASIC

K-BASIC under OS-9 and FLEX will compile TSC BASIC, XBASIC and XPC Source Code Files.

$AVE $100

K-BASIC now makes the multitude of TSC XBASIC Software available for use under OS-9. Transfer your favorite BASIC Programs to OS-9, compile them, Assemble them, and BINGO -- useable, multi-precision, familiar Software is running under favorite Operating System!

!!! SPECIAL ~~$199.00~~ $99.00 !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE

With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts **Features**

Sculptor for 68020 OS-9 & UniFLEX $995

DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Packed, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

–	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
–	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

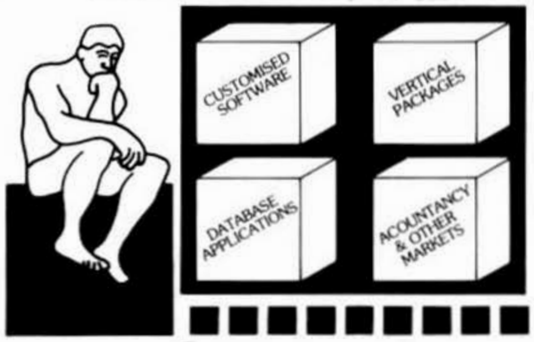
PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type



* Full Developement Package
** Run Time Only
*** C Key File Library

OS-9/UniFLEX
IBM PC Zenix
MS DOS Network -- $995 / $199 / $498
* * ***

68000 UniFLEX
Altos Zenix
UNIX -- $1595 / $319 / $798
* ** ***

MS DOS
PC DOS -- $595 / $119 / $595
* ** ***

MUSTANG-020™ Users - ask for special discount.

Sculptor is a Trademark of Microprocessor Developments Ltd.

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends-

F - FLEX, CCF - Color Computer FLEX
O - OS-9, CCO - Color Computer OS-9
U - UniFLEX
CCD - Color Computer Disk
CCT - Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

South East Media
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler: extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00

DYNAMITE+ — Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00 - CO, Obj. Only $ 59.95
F, " " $100.00 - O, object only $150.00
U, " " $300.00

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

F, CCF - $198.00

PASC from S.E. Media - A Flex9 Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.

FLEX $95.00

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers.* "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer, etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

F and CCF - $195.00

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

F and CCF - $295.00

C Compiler from Introl — Full C except Doubles and Bit

(615)842-4600 Telex 5106006630

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.

FLEX, CCF, OS-9 (Level II ONLY), U - $575.00

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $99.95 F 8" - $99.95

PASCAL Compiler from OmegaSoft (now Certified Software) -- For the *PROFESSIONAL;* ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.

F and CCF - $425.00 - One Year Maint. $100.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

FLEX, CCF, OS-9 Compiler /Assembler $199.00

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*

FLEX, CCF; Normally $199.00
Special Introductory Price $99.95

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility. Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - $58.95

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems

FOR 6809 FLEX-SK-DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced format Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends

F - FLEX, CCF - Color Computer FLEX
O - OS-9, CCO = Color Computer OS-9
U - UniFLEX
CCD - Color Computer Disk
CCT - Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

Telex 5106006630

(615)842-4600

South East Media

6900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

ITS EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...). The possibilities are unlimited...

FOR 6809 FLEX-SK-DOS(5/8") **$249.95**

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, CCF - $99.95

Macro Assembler for TSC -- The FLEX STANDARD Assembler.
Special -- CCF $35.00; F $50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX.
FLEX, CCF, OS-9 $99.00

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, CCF $150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
F, CCF - $75.00

XMACE -- MACE w/Cross Assembler for 68 00/1/2/3/8 *F, CCF - $98.00*

CROSS ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
68000 or 6809, FLEX, CCF, OS-9, UniFLEX
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - source $300.00

XASM Cross Assemblers for FLEX from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.
Complete set, FLEX only - $150.00

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.
FLEX, CCF, OS-9/6809 $249.00

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
O & CCO obj. only - $89.95

Lucidata PASCAL UTILITIES (Requires LUCIDATA Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
F, CCF -- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media -- A **UniFLEX BASIC decompiler** Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
U - $219.95

LOW COST PROGRAM KITS from Southeast Media -- The following kits are available for FLEX on either 5 or 8 inch disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX UTILITIES KIT $39.95**
   CATS.CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization where required
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer program
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 Regular $149.95
SPECIAL INTRODUCTION OFFER *$69.95*

☎ (615)842-4600 Telex 5106006630
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

DISK UTILITIES

OS-9 VDisk from S.B. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX Directory, Delete FLEX Files, Copy both directions, etc. FLEX users use the special disk just like any other FLEX disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

HIER from S.B. Media - *HIER is a modern hierarchal storage system for users under FLEX.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX disk (8 - 5 - hard disk) can have sub directories.** By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. Each **directory looks to FLEX like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
** Introduction Special ** *$69.95*

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends--
F - FLEX, CCF - Color Computer FLEX
O - OS-9, CCO - Color Computer OS-9
U - UniFLEX
CCD - Color Computer Disk
CCT - Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

Telex 5106006630
(615)842-4600
South East Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

COPYMULT from S.E. Media – Copy LARGE Disks to several smaller disks. FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
Completely documented Assembly Language Source files included.
ALL 4 Programs (FLEX, 8" or 5") $99.50

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
F and CCF 5" - $50.00 F 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight task on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between task at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
O & CCO - obj. only - $49.95

FLEX DISK UTILITIES from Computer Systems Consultants - - Eight (8) different Assembly Language (w/ Source Code) FLEX Utilities for every FLEX Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
ALL Utilities include Source2 (either BASIC or A.L. Source Code).
F and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. **This is one of the strongest CHESS programs running on any microcomputer,** *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F and CCF - $79.95

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
FLEX, CCF, OS-9, UniFLEX, 68000 & 6809 with Source $100.00 - without Source $50.00

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transfered between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks)	*$99.95*
X-TALK Software (2 disks only)	*$69.95*
X-TALK with CMODEM Source	*$149.95*

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
U - $299.99

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.
* Now supplied as a two disk set:
Disk #1: JUST2.CMD object file, JUST2.TXT PL9 source:FLEX - CC
Disk #2: JUSTSC object and source in C: FLEX - OS9 - CC
The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!
Disk (1) - PL9 FLEX only - F & CCF - $49.95
Disk Set (2) - F & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.B. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX $129.50
• SPECIAL INTRODUCTION OFFER • $79.95
SPECIAL PAT/JUST COMBO (w/source)
FLEX $99.95
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00
Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

Regular $129.95
SPECIAL INTRODUCTION OFFER FLEX $69.95

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, STAR-DOS Regular $69.95
Limited Special Offer: $39.95

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

SPELLB "Computer Dictionary" from S.E. Media -- **OVER 150,000 words!** *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/STAR-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,
F or O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,
F or O - $99.95, U - $149.95

(615)842-4600 Telex 5106006630
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES6809 CCF and CCO - $59.95,
F or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

F or O - $329.95, U - $549.95
O, 68000 $595.00

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.

F, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

F - $59.95, U - $89.95

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends—

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

** Shipping **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

VIRTUAL TERMINAL REVIEW

A few years back I was at a small computer show. One of the exhibitors was demonstrating his system. He was running one of those slick demo programs that produce game room graphics. "Right now this one terminal is running 8 tasks," the demonstrator bragged.

"Well I can do that on my home system, under OS-9," I retorted.

"Ah! But can you do this?" he asked. He touched a few keys and the graphics was gone. He was in a mail program, reading his mail. A few keys touched again, and he was in a word processor, editing some text. He touched the keyboard again, and a program was just finishing being compiled.

"Ah, that's o.k. I've seen enough," I said, stopping him. I couldn't do that.

"Ah, but I've got 4 more to go!"

"That's quite alright," I said. I backed down. OS-9 can definitely multitask, but I couldn't do what he was doing. Darn!

Now the good news. South East Media has come up with a sharp package called VIRTUAL TERMINAL. It allows an OS-9 user to start as many as 8 tasks from one terminal. They can all be shells. Or maybe one is a word processor, another a compiler and still another a database. And you'll still have 5 left. The great thing is the user can move freely between them whenever he wants.

Hands on experience is the best educator. VIRTUAL TERMINAL comes with a file called VMOD. This is installed using the load command. Entering:

SWITCH 0 I

and Poof! you're in the virtual terminal /V0. Plus there is another one available. Type the Tilde (that is the little ~) and the number 1. And you're now in terminal /V1. With the Tilde you can toggle back and forth between them. Remember this is only a demonstration. There are still 6 terminals left.

You can now start something on one terminal and then switch to the other. Do a few things on it and then move back to the first. I started Basic09 on /V1, entered into the editor mode and typed the Tilde and 0. Instantly, I was in the OS-9 shell on /V0. I entered PROCS and saw that Basic09 was indeed still running, only asleep. I entered the Tilde and 1. Back I went into the Basic09 editor, just like I never left it. I left Basic09 and typed the following:

ECHO HELLO VIRTUAL TERMINAL NUMBER 0 >/V0

I typed ~ and 0. I arrived in /V0 and so did my message. Enough of this! I think you get the idea.

As you already have noted, the ~ is used to indicate some course of action. It is an indicator that what follows effects the virtual terminal you're in. It was picked, since it probably is one of the least used characters from the keyboard. Here is a listing of what you can change.

```
-O  Remove output sequence and inhibit screen re-
print
O   Initialize use of output sequence and screen re-
print
-I  Remove input sequence
I   Initialize use of input sequence
/   Display current device
?   Display available devices
-c  Change switch character to whatever c is
n   Change to terminal n
Q   Quit
```

A number after the switch character moves you to that terminal. Entering a / will print the virtual terminal number in use. A ? will print all the available terminals and their numbers. There are ASCII character streams that are sent to the terminal and process when the a terminal is entered. The output sequence is sent to the physical terminal when entering the virtual terminal. The input sequence is sent to the running process when entering the virtual terminal. These can be switched on and off. And the switch character (~) can even be changed.

There are a number of modules that make Virtual Terminal possible. / V0 through /V7 are the device descriptors. The source is included, so you can modify these for your system. VTERM is the device driver. It uses SCFMAN as its file manager. There is Virtual. It is the data module for each terminal. The lowercase n is the virtual terminal number. Virtual is created with a program called CONFIG. I'll tell you more about it in a minute. There is VRTGO, which is similar to SYSGO. It starts the shell for a virtual device. And finally is SWITCH. It is the supervisor. It handles input from the physical terminal and passes it to a data area. Or if it is an option switch, it effects the desired action.

So, what about CONFIG? It is used to setup the data modules for the terminals. It helps you customize your virtual terminals. It allows you to setup output sequences for your particular terminal, input sequences for whatever process you are running, the startup process and a bunch of other things. CONFIG is menu driven. It is quick and easy to use.

There are other subtlety about virtual terminal. When switching from one terminal to the next, you may not want to have the process go to sleep. This can be altered by changing the TYPE statement in the descriptor. Say you've got a compiler running. It takes some time. With Virtual Terminal, start your compiling, and switch to another terminal. Come back later when things are finished. With a little ingenuity, have a message sent to the terminal your on, when compiling is done.

Here is another thought. Have you ever wanted to monitor another process? It can be done. Once a friend you was showing me a new program. He started it one terminal. We darted across the room to another terminal. Used PROCS and whatever else he had to examine what was happening. Went back to the first entered some input. Went to the second, and observed some more. Back to the first and so forth. It was interesting. I personally cannot afford more then one terminal. And don't enjoy marathon terminal hopping. With VIRTUAL TERMINAL it is easy. Switch back and forth without ever getting up and use only one terminal.

VIRTUAL TERMINAL is real winner for everything you do. If you have ever thought about having a spare terminal. Or you want to take full advantage of OS-9's multi-tasking capability. VIRTUAL TERMINAL is what you're looking for.

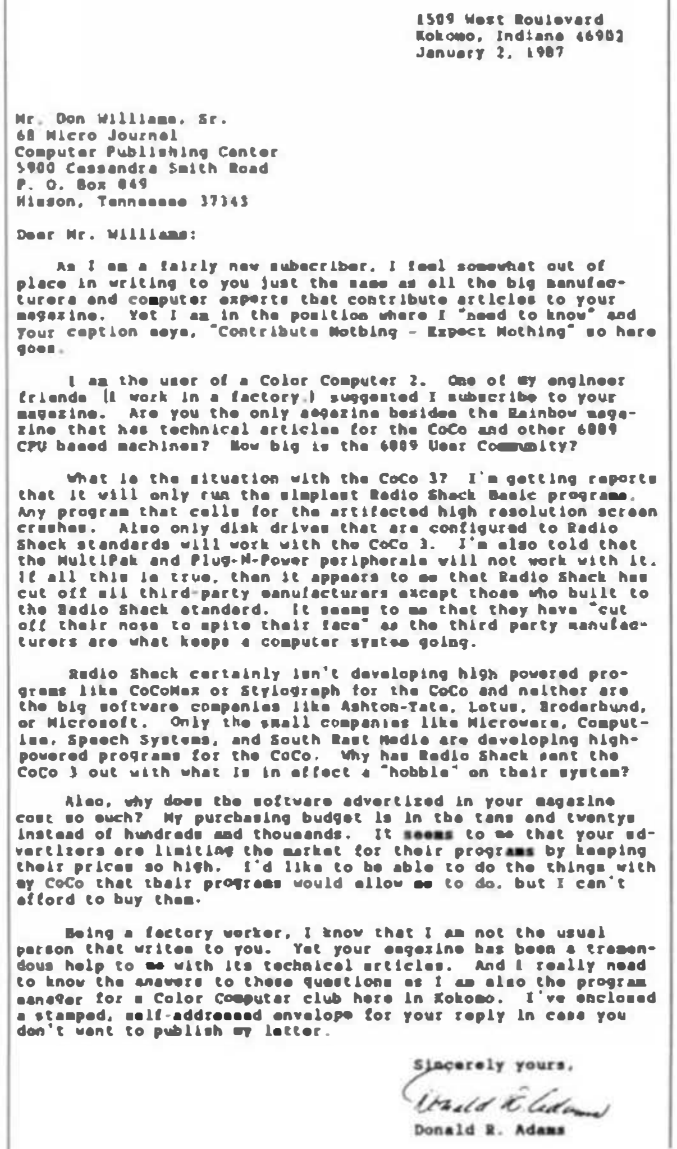
VIRTUAL TERMINAL
Regular $79.95
Limited Time ONLY $49.95
S.E. MEDIA

See Catalog - Center Section

Bit-Bucket

By: All of us

"Contribute Nothing - Expect Nothing", DMW '86



1509 West Boulevard
Kokomo, Indiana 46902
January 2, 1987

Mr. Don Williams, Sr.
68 Micro Journal
Computer Publishing Center
5900 Cassandra Smith Road
P. O. Box 849
Hixson, Tennessee 37343

Dear Mr. Williams:

As I am a fairly new subscriber, I feel somewhat out of place in writing to you just the same as all the big manufacturers and computer experts that contribute articles to your magazine. Yet I am in the position where I "need to know" and Your caption says, "Contribute Nothing - Expect Nothing" so here goes.

I am the user of a Color Computer 2. One of my engineer friends (I work in a factory.) suggested I subscribe to your magazine. Are you the only magazine besides the Rainbow magazine that has technical articles for the CoCo and other 6809 CPU based machines? How big is the 6809 User Community?

What is the situation with the CoCo 3? I'm getting reports that it will only run the simplest Radio Shack Basic programs. Any program that calls for the artifacted high resolution screen crashes. Also only disk drives that are configured to Radio Shack standards will work with the CoCo 3. I'm also told that the MultiPak and Plug-N-Power peripherals will not work with it. If all this is true, then it appears to me that Radio Shack has cut off all third-party manufacturers except those who built to the Radio Shack standard. It seems to me that they have "cut off their nose to spite their face" as the third party manufacturers are what keeps a computer system going.

Radio Shack certainly isn't developing high powered programs like CoCoMax or Stylograph for the CoCo and neither are the big software companies like Ashton-Tate, Lotus, Broderbund, or Microsoft. Only the small companies like Microware, Computize, Speech Systems, and South East Media are developing high-powered programs for the CoCo. Why has Radio Shack sent the CoCo 3 out with what is in effect a "hobble" on their system?

Also, why does the software advertized in your magazine cost so much? My purchasing budget is in the tens and twentys instead of hundreds and thousands. It seems to me that your advertizers are limiting the market for their programs by keeping their prices so high. I'd like to be able to do the things with my CoCo that their programs would allow me to do, but I can't afford to buy them.

Being a factory worker, I know that I am not the usual person that writes to you. Yet your magazine has been a tremendous help to me with its technical articles. And I really need to know the answers to these questions as I am also the program manager for a Color Computer club here in Kokomo. I've enclosed a stamped, self-addressed envelope for your reply in case you don't want to publish my letter.

Sincerely yours,

[illegible]

Donald R. Adams

Editor's Note: Thanks Don for the letter and kind words about 68 Micro Journal. Like you, I get some pretty hairy reports about the CoCo III, also I get other inputs from some fairly happy users. So, I can only go on what I hear, as I am not as up on the III as the II.

My complaint is that Tandy doesn't support it to the level I think it deserves. I suggest you read my 'RAMBLINGS' column, this issue.

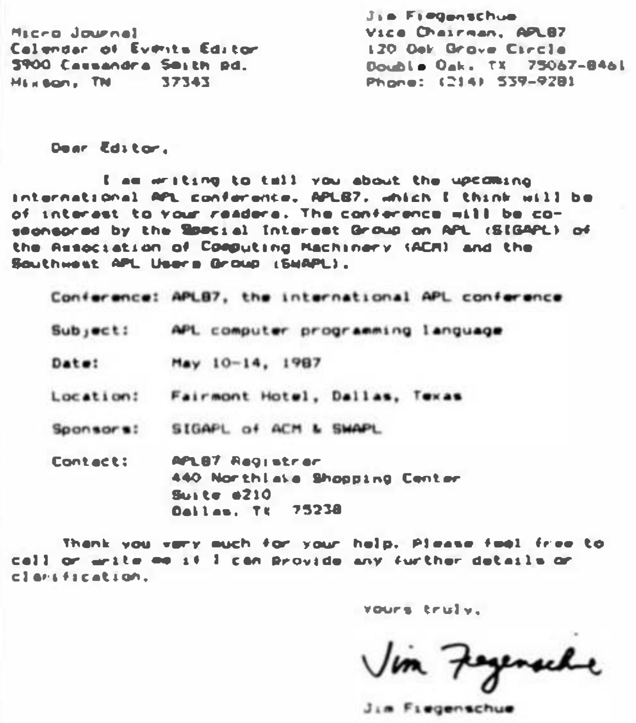
I do not believe that any of the larger software vendors will ever do anything for the CoCo (any version) as the CoCo market is a low budget type. Also, to get support from most vendors it requires some cooperation from the manufacturer. I have already expressed myself on that subject.

You might bear in mind that if the CoCo had been done 'full-bore', it with good software, would run circles around many other Tandy products. Not too good from the profit angle. A lot more profit in selling the expensive stuff!

The reason software cost so much is that it's cost is based on two primary factors - cost of development and production and number of unit sold at any particular prise. So, the more units sold, the less you have to charge to stay in business. And you have to bear in mind, most all the good 6809 software sold is better than most of the stuff for other CPUs. A fact! And, if you will notice there are a lot of bargains in S.E. Media's catalog. The stuff they have control of is all sold at very reasonable prices, and with source in most cases. However, the price others put on their product is beyond S.E. Media's control. But they do try - and hard to bring it to you at rock-bottom prices. Without S.E. Media and 68 Micro Journal where would the serious 68XXX user go?

Oh, another point, Microware isn't too small a company anymore. They did it right and now are foremost in 68XXX software sales and distribution. Others fell by the way-side due to customer and product neglect. Microware isn't perfect, but they try. Just watch OS-9 and CD-I blossom in the next few years.

DMW



Micro Journal
Calendar of Events Editor
5900 Cassandra Smith Rd.
Hixson, TN 37343

Jim Fiegenschue
Vice Chairman, APL87
120 Oak Grove Circle
Double Oak, TX 75067-8461
Phone: (214) 539-9281

Dear Editor,

I am writing to tell you about the upcoming international APL conference, APL87, which I think will be of interest to your readers. The conference will be co-sponsored by the Special Interest Group on APL (SIGAPL) of the Association of Computing Machinery (ACM) and the Southwest APL Users Group (SWAPL).

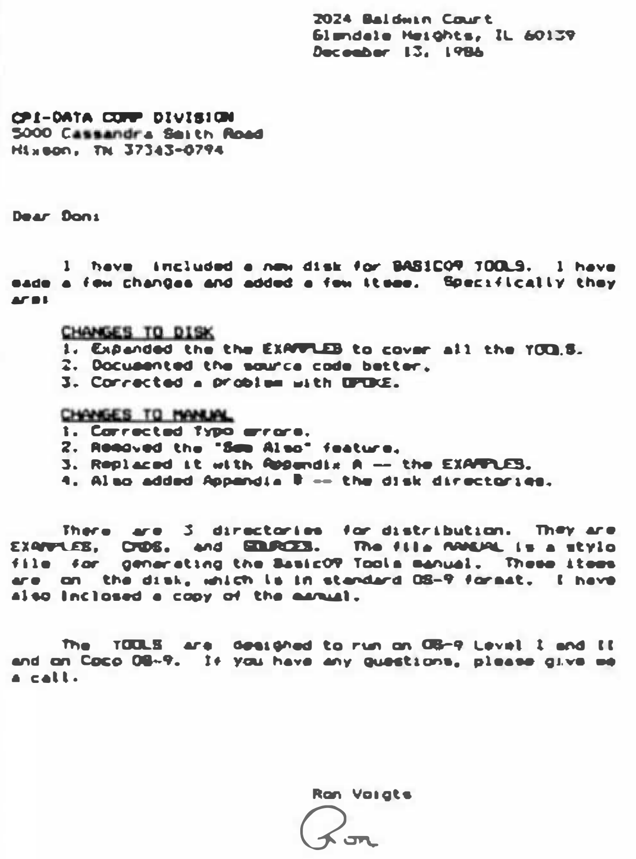
Conference:	APL87, the international APL conference
Subject:	APL computer programming language
Date:	May 10-14, 1987
Location:	Fairmont Hotel, Dallas, Texas
Sponsors:	SIGAPL of ACM & SWAPL
Contact:	APL87 Registrar 440 Northlake Shopping Center Suite #210 Dallas, TX 75238

Thank you very much for your help. Please feel free to call or write me if I can provide any further details or clarification.

Yours truly,

[illegible]

Jim Fiegenschue



2024 Baldwin Court
Glendale Heights, IL 60139
December 13, 1986

CPI-DATA CORP DIVISION
5000 Cassandra Smith Road
Hixson, TN 37343-0794

Dear Don:

I have included a new disk for BASIC09 TOOLS. I have made a few changes and added a few items. Specifically they are:

CHANGES TO DISK
1. Expanded the the EXAMPLES to cover all the TOOLS.
2. Documented the source code better.
3. Corrected a problem with DPOKE.

CHANGES TO MANUAL
1. Corrected Typo errors.
2. Removed the "See Also" feature.
3. Replaced it with Appendix A -- the EXAMPLES.
4. Also added Appendix B -- the disk directories.

There are 3 directories for distribution. They are EXAMPLES, CMDS, and SOURCES. The file MANUAL is a stylo file for generating the Basic09 Tools manual. These items are on the disk, which is in standard OS-9 format. I have also inclosed a copy of the manual.

The TOOLS are designed to run on OS-9 Level I and II and on Coco OS-9. If you have any questions, please give me a call.

Ron Voigts

Ron

Don Williams Sr., Publisher
'68' Micro Journal
POB 849
5900 Cassandra Smith Road
Hixson, Tennessee 37343

Dear Sir:

I wish to share with your readers a project that I undertook to make further use of my computer system. Perhaps some of the ideas will be of interest to other readers. I have an SS-50 computer (SWTP 69/K) with SSB's DCB4-A disk controller, DOS69D operating system and two 8 inch drives. I worked in Canada's arctic and lived in Calgary. An SS-50 system is not very suitable for packing around.

I dialed into the 68 Micro BBS one time and realized I could do the same with my system.

The company that I worked for in the Arctic used Apple II computers expanded with Z80 cards and CP/M as data communication terminals. A direct telephone line was used between the Arctic and the office in Calgary via satellite. Surely I could access my computer remotely via the same Apple terminals.

First step, obviously, is that the computer has to boot automatically into "DOS". I was using the SSB monitor rom "MON09D". I disassembled the code with the help of the manual supplied with the monitor (the code in the ROM turned out to be slightly different than what was described in the manual). I found the "monitor soft start" address and then a few lines later is "JSR INEEE", where the monitor sits in a loop waiting for input from the keyboard. I changed that to "LDA #$51" followed by a "NOP" to fill out the space left by the removal of the "JSR INEEE". The "LDA #$51" is load immediate the hex value corresponding to the letter "Q". "Q" is the keyboard entry for "cold boot" in "MON09D". The code that follows in the monitor is the command lookup routine that looks for a match in the command table. This isn't changed. I burned this change into a 2716 EPROM. and put the modifed chip in place of "MON09D" on the CPU board. With this now when the computer is turned on or when "RESET" is hit it automatically boots up "DOS". I've tried this with SWTP monitor "SBUG-E" and it also works with it also. I suspect most monitors could be modified to auto-boot FLEX, STAR-DOS etc.

Now to look at my modem. On a visit to the "STATES" I picked up a "US ROBOTICS PASSWORD" just when it was being introduced to the market. The specifications indicated the modem supplied "CARRIER

DETECT" on pin 8 of the DB 25 connector and also "SPEED INDICATE" on pin 12. Very useful. I made up a little circuit on a piece of perf board containing a 74121 configued as a "ONE SHOT" monostable multivibrator. I stuck the perf board on the back of the CPU board with double sided sticky tape and picked up +5 volts and ground from the CPU board. The output of the "ONE SHOT" is across the reset terminal of the CPU board. I ran a wire from the input of the "ONE SHOT" over to the serial board to pick up "CARRIER DETECT" on pin 8 of the "RS-232" connector. I used an inline "MOLEX" connector so I could open the connection when I wish to work on the CPU or other cards. What happens with this mod is when someone dials into the computer the modem answers the telephone and sends a tone down the telephone line. The other end responds and sends his tone back. When the modem recognizes the tone it asserts the "CARRIER DETECT" pin. This toggles the "ONE SHOT" and it toggles the "RESET" briefly on the CPU card. This cold boots the computer.

I have both SWTP's SBUG-E and SSB's MON09D and they have a bit of incompatibility with each other in where they expect the address of the system console. Mainly SBUG-E expects the console at $E004 and MON09D at $E008. I use "DATA SYSTEMS 68" dual serial card in slot 0 as it recognizes both $E004 and $E008 as being the same thing (obviously incomplete address decoding). This lets me use both MON09D and SBUG-E without having to make any hardware changes. MON09D uses a 2 byte vector at $DFE2 called "CPORT" that holds the actual address for the console port. This means I can switch my console port between my normal port at slot 0 and my modem port at slot 2 merely by poking the appropiate address into "CPORT". The MP-S2 serial card has an additional input called the "CONTROL LINE INPUT REGISTER" addressed at $XXXE and $XXXF. This is a 74LS240 octal buffer chip which reads a 1489 quad RS232 receiver chip. I put a wire jumper from pin 12 on the DB-25 connector (the SPEED INDICATE from the modem) to one of the inputs of the 1489 receiver (called IN1 on SWTP's MP-S2). Then in my software I read $E02F and initialize the ACIA clock to X16 (1200 baud) or X64 (300 baud). This means the jumper on the MP-S2 card must be set to 1200 baud. This means my system will answer at either 300 baud or 1200 baud depending if the caller uses "BELL 103" or "BELL 212" tones. "DOS69D" has a startup routine like "FLEX" and "STAR-DOS" called "START.UP". I have a disk dedicated to the "BBS" with an appropiate start up file, the first command on the start up file is called "SETSPD". This program loads the appropiate port address into "CPORT", picks up the speed and intializes the modem ACIA and prints the message string "STANDBY - BOOTING UP SYSTEM" and then reloads "CPORT" to the normal console address to continue the start up sequence. The start up file then does a SET command (something like "ASN" in "FLEX") followed by "DV.RCTL" command (which tells "DOS69D" the parameters of the disk drive controller card installed in the system) and by the "LUNCTL" command (which tells "DOS69D" the parameters of all the disk drives I have in the system) followed by RUN TIME.SYS which installs my clock driver program, and finally I switch "CPORT" back to the modem port and run my "brag tape" (amateur radio reference to a program that describes my system to the caller and a brief description of how to use the system). Then it drops into SSB's prompt "DOS" and is ready for use.

Here is what my start up file looked like when I put my system on line.

```
SETSPD
   SET,CRT=E024,STOP=13,CONT=11,HC=E044,
NULL=03,DP=50,WD=72,MEMAX=97FF,WK=01,
DATE=TH OCT 08 1983
  DVRCTL.UPDATE,0/UM=F0
   LUNCTL.INSTAL,1=0,5/8/DD/SS/SR=0/NAME=WK
   LUNCTL.INSTAL,2=0,6/8/DD/SS/SR=0
  RUN,0:TIME.SYS
 SETERM
```

"SETERM" is my "brag tape". The "STOP=13" in the SET command means output can be halted with a control S and "CONT=11" is control Q. This means my system can be controlled with X-ON/X-OFF protocol. The other entries in SET is really for my teletype machine that I had at $E044. The reason for the two LUNCTL entries, I configued one of my 8 inch double sided drives as two single sided drives. It made my system look like it had more drives on it than it really had. I've changed this startup file since to reflect later changes.

I called this "The 6809 System" and had it running for a year and a half. It was on-line only part of the time (when I was in the arctic) and I did not advertise the telephone number, the system was primarily for my own use, though I have given the telephone number to a few friends and it did get used by some radio amateurs in the Calgary area as a mailbox. Since that time I had added "STAR-DOS" to my system and later added Peripheral Technology's FD-2 disk controller board and two 5 inch drives and now I can switch between "DOS69D" on the 8 inch drives and "STAR-DOS" on the 5 inch drives. I took the system off-line early last spring when I finished working in the Arctic. By the way, Apple II keyboards are woefully deficient in some characters, like " ", " { ", " } ", " [", "] ", and " _ ". This made it a royal pain to try to write code in PL/9, assembler and C from the arctic on my system.

I hope the description of my system has been of interest to your readers and may contain a few ideas for some on how the computers can be adapted for various uses.

Yours truly,

Walter Isaacson
#19, 1614 - 22 Ave., S.W.
Calgary, Alberta

```
                        * SETSPD.1
                        * This program checks the type of call
                        * (Bell 103 or Bell 212) as received
                        * by the modem and adjusts the baud
                        * rate of the 6850 acia.
                        *
                        * The 6809 System
                        * Walter Isaacson
                        * #19 1616 - 22 Ave. S.W.
                        * Calgary, Alberta T2T 0R8
                        * Tel (403) 245 5098
                        *
               E024     MODPRT  EQU   $E024    This is where the modem lives
               E004     KEYPRT  EQU   $E004    Normal terminal port
               DFE2     CPORT   EQU   $DFE2    Control Port address
                        *
                        * DOS690 EQUATES
               D2A6     PSTRNG  EQU   $D2A6    Output a string
               D203     WARMS   EQU   $D203    Warm start
                        *
CD00                            ORG   $CD00    Transient command area
CD00 B6    E02F         START   LDA   $E02F    Control line input register
CD03 84    02                   ANDA  #$02
CD05 81    02                   CMPA  #$02
CD07 27    0C                   BEQ   X16
CD09 86    03                   LDA   #$03
CD0B B7    E024                 STA   MODPRT
CD0E 86    16                   LDA   #$16     64/clock
CD10 B7    E024                 STA   MODPRT
CD13 20    0A                   BRA   MESSAG
CD15 86    03           X16     LDA   #$03
CD17 B7    E024                 STA   MODPRT
CD1A 86    15                   LDA   #$15     16/clock
CD1C B7    E024                 STA   MODPRT
CD1F CC    E024         MESSAG  LDD   #MODPRT
CD22 FD    DFE2                 STD   CPORT
CD25 8E    CD3A                 LDX   #MESS
CD28 BD    D2A6                 JSR   PSTRNG
CD2B CC    FAC3                 LDD   #$FAC3
CD2E FD    DFxx                 STD   SWVEC    NMI vector
CD31 CC    E004                 LDD   #KEYPRT
CD34 FD    DFE2                 STD   CPORT
CD37 7E    D203                 JMP   WARMS    and return to DOS690
CD3A 00 0A 0A           MESS    FCB   $00,$0A,$0A
CD3D 2A 20 53 54                FCC   '* STANDBY - BOOTING UP SYSTEM *'
CD41 41 4E 44 42
CD45 59 20 2D 20
CD49 42 4F 4F 54
CD4D 49 4E 47 20
CD51 55 50 20 53
CD55 59 53 54 45
CD59 4D 20 2A
CD5C 00 0A 00                   FCB   $00,$0A,0
                                END   START
```

EOF

148 Winkler Rd.
E. Windsor, CT. 06088

Mr. Don Williams
68 Micro Journal

Dear Don,

I would like to comment on two programs I purchased this year, as well as present you with a new PL9 core program that I have been updating.

Recently I purchased Research Instruments Co. BAS-EDIT program and had serious problems installing it on a non-ACIA system. Since the program AFFIX.CMD was suppose to handle this situation, and the source was provided, I am sending you the changes necessary to make it work properly (see AFFIX.SRC on this disk).

The basic problems in AFFIX.CMD have to do with two errors in the testing of the ACIA port. If any other number other than a "9" is returned to the software, it will crash. Also, if the system I/O is not supported by an ACIA, the software will hang. To avoid this problem, the following changes should be made to the source code:

```
START2  LDX #MSG1

        JSR PDATA

        LDX #$E004

        BRA CA1     DELETE THIS LINE
```

Add the following code:

```
START3  JSR STAT

        BEQ START3

        JSR RAW4

        BRA CA2
```

Where:

1. Label CA2 is the CMPA #'9 statement after the label CA1.
2. Label RAW4 is the LDA 1,X statement after the label RAW1 in the RAWIN routine.
3. The Rawin routine has two PSHB statements and no PULB statement! The second PSHB should be changed to PULB.

Last, I have a program that may be of interest to your readers. It is the core for a MULTITASKER that I have written and use in a multiprocessor controller. It is similar to the multitasker example in Windrush's PL9 users manual, with the exception that the user has control over the individual task duration, frequency of entry, and the ability to enable or disable individual tasks. A task swap lock has also been included to help program functions that are sensitive to task swapping.

I have generously commented the source code, and have included what I call DUMB_TSK.LIB as an example of usage. The only procedures that are hardware dependent are related to the system clock I/O, and one memory location used to store the Y reg so globals can be used by the interrupt routine. Both are commented so the user should clearly understand how they are used.

I am not going to get involved with a description of a multitasking environment. DUMB_TSK.LIB should be read carefully so that the interactions between terminal related tasks can be recognized and used in further software development. In the controller where this is used, a central processor (MC6809) is communicating with four other MC6809's that EACH communicate with 128 other custom processors. The multitasker core is used to handle communications to the four MC6809's, a terminal, printer, and a remote control box. The tasker is also responsible for activation of all system control functions that are time dependent. Needless to say, there is a lot of task interaction, especially where more than one task uses the same system hardware (i.e., the terminal and printer). Even with all this activity, there is ample time to do a scanning process of a matrix keypad! (See "Keyscan", May '68 Micro Journal, page 40.) Without PL9, this project would have been doomed!

Please feel free to either print this program, or include it in the reader service library. I hope that the above fixes and this program help you and your readers. Keep up the great work on a great magazine.

Fred Stucklen

EOF

```
/* Clock I/O Routines          Version 1.00          7-Nov-86        fws

/* The Real Time Clock is used to generate periodic interrupts.          */
/* The user is warned that the clock MUST be set up properly for         */
/* the particular system that is being used. This library should be      */
/* configured to your particular system hardware address. If you         */
/* don't have any LED's that can easily be turned on and off, set        */
/* the LED address AT statement to an unused memory location.            */
/*   This library used the MC146818 Real Time Clock. The interrupt       */
/* rate is set to approximately 4 ms. Other timers types can just as     */
/* easily be used if the appropriate support software is written.        */
/* I have included AT statements for the PT-69 system.                   */


constant IRQRATE = $06;            /* 146818 RTC IRQ Rate      */
constant CRYSTAL_FREQ = $20;        /*    4.194304 Mhz=$00  (PT-69)  */
                                    /*    1.048576 Mhz = $10         */
                                    /*    32.768   Khz = $20 (CG2 Sys) */

/* AT $E01C:byte clk_data,clk_adr;              /* MC146818 RTC  PT-69      */
/* AT $E000:byte led(0), led0;                  /* Diagnostic LED's         */

  AT $F720:byte clk_data,clk_adr;  /* CG2 Dev system clock adress      */
  AT $F778:byte led(0), led0;      /* CG2 Dev system diagnostic led's */

procedure clk_irq_on;                  /* Turn on clock IRQ's          */
   clk_adr=$0B;
   clk_data=clk_data or $40;
endproc;

procedure clk_irq_off;                /* turn off clock IRQ's         */
   clk_adr=$0B;
   clk_data=clk_data and $BF;
endproc;

procedure get_clk(byte adr):byte data;  /* Read a clock register        */
   clk_adr=adr;
endproc clk_data;

procedure put_clk(byte adr,data);       /* Write to a clock register  */
   clk_adr=adr;
   clk_data=data;
endproc;

procedure clock_init: byte dat;
   put_clk($0B,(get_clk($0B) or $10)); /* Disable while initing */
   put_clk($0A,(CRYSTAL_FREQ or IRQRATE));
   put_clk($0B,(get_clk($0B) and $7B)); /* [illegible] and $7b...          */
endproc;

procedure clear_clock_IRQ;
   clk_adr = $0C;                    /* Read Reg C to clear IRQ   */
   acca = clk_data;
endproc;


   global.lib

/* Multi-Tasker variable definitions       ver 1.0  (7-Nov-86)   fws  */


/* Task stacks allocations. The size can be modified as necessay by   */
/* the user, but should be capable of handling all task local          */
/* variables, as well as the local variables in routines called by    */
/* that task. No stack overflow checking is performed....             */
/* There should a stack for each of the tasks in the system (In this */
/* example there are 6). The subscript in the globals listed below    */
/* should be equal to the total number of stacks used.                */

global integer stack0(256),stack1(256),stack2(256),stack3(256),
               stack4(256),stack5(256),
               old_stack,       /* old stack, used in add/del */
               stack_ptr(6),    /* stack pointers */
               stack_adr(6)     /* stack address */

     : byte    task_stat(6),    /* task stack status: ACTIVE, INACTIVE */
               .stack_ccr,      /* task stacks ccr, used to add tasks */
               cur_task,        /* current task number */
               active_tasks,    /* number of active tasks */
               new_task,        /* new task number, used in add/del */
               task_skips(6),   /* running skip count from task_freq */
               slice_cnt,       /* running slice count, max =    */
               task_freq(6),    /* Max skip rate for each task   */

               swap_lock,       /* flag, locks out swapping */
               num_swaps,       /* number of task swaps */

               erflag, keychar, /* Hexio.lib globals */

               key,             /* Task_1 example temp key value */
               keystat,         /* task_1 example key status     */
               dumb_cnt(6),     /* Task counters                 */
               term_stat;       /* Terminal in use flag...       */

/* Please leave this copywrite notice intact.                          */

byte copyright "Multi-Tasking Kernel software copyright 1986";
byte notice    "Version 1.0 7-Nov-86";
byte author    "written by: F.W.Stucklen";


/* SLICE_MAX can be redefined by the user if necessary. It represents  */
/* for a particular task,ie, at SLICE_MAX=5 and the timer interrupt    */
/* rate at 4ms results in a 20ms duration for each unit value.         */
/* The value of task_freq(task number) represents how many cycles      */
/* time the particular task will be skipped. Therefore actual task     */
/* execution time is a function of the number of active tasks,         */
/* and their task_freq.                                                */

constant  ACTIVE=$01,  INACTIVE=$00,  LAST_TASK=$05,   SLICE_MAX = $05,
          TRUE = -1,   FALSE = 0;

byte bmask  $01,$02,$04,$08,$10,$20,$40,$80;



/*  SYSTEM  CONSOLE  INPUT/OUTPUT  ROUTINES   V:4.00  */


constant nul = $00,
         etx = $03,
         bel = $07,
          bs = $08,
          lf = $0a,
          cr = $0d,
         can = $18,
         esc = $1b,
          sp = $20;


procedure monitor;
  gen $6e,$9f,$d3,$f3;  /* JMP [$D3F3]  ('MONIT') */
endproc;


procedure warms;
   jump $cd03; /* FLEX WARM START ENTRY POINT */
endproc;


procedure getchar;
   call $cd15; /* FLEX 'GETCHR' */
endproc acca;
/* how many timer interrupts are allowed per unit of allicated time     */
 procedure getchar_noecho;
    gen $ad,$9f,$d3,$e5;  /* JSR [INCHNE] */
 endproc acca;


 procedure getkey;
    call $cd4e; /* FLEX 'STAT' */
    if ccr and $04  /* IMPLICIT <> 0 */
       then acca = nul;
       else getchar_noecho;
 endproc acca;


 procedure convert_lc(byte char);
    if char >= 'a .and char <= 'z
       then char = char - $20;
 endproc char;


 procedure get_uc:byte inchar;
 endproc convert_lc(getchar);


 procedure get_uc_noecho:byte inchar;
 endproc convert_lc(getchar_noecho);


 procedure putchar(byte char);
    acca = char;
    call $cd18; /* FLEX 'PUTCHR'  (HONOURS 'TTYSET' PARAMETERS) */
 endproc;
```

```
procedure printint(integer n);
   if n < 0
      then begin
              putchar '-;
              n = -n;
           end;
   if n >= 10 then printint n/10;
   putchar n\10 + '0;
endproc;

procedure remove_char;
   putchar(bs);
   putchar(sp);
   putchar(bs);
endproc;


procedure input(byte .buffer,length):byte char:integer pos;
   pos = 0;
   repeat
      char = getchar_noecho;
      if char
         case bs
            then begin
                    if pos > 0
                       then begin
                               remove_char;
                               pos = pos - 1;
                               end;
                       else putchar(bel);
                    end;
         case can
            then begin
                    while pos > 0
                       begin
                          remove_char;
                          pos = pos - 1;
/* how many timer interrupts are allowed per unit of allicated time     */
                       end;
                   end;
      if char >= sp
         then if pos < length
                 then begin
                         putchar(char);
                         buffer(pos) = char;
                         pos = pos + 1;
                      end;
                 else putchar(bel);
   until char = cr;
   buffer(pos) = 0;
endproc .buffer;


procedure crlf;
   putchar(cr);
   putchar(lf);
endproc;


procedure print(byte .string): byte char;
   while string /* IMPLICIT <> 0 (NULL) */
      begin
         if string = '\
            then begin
                    .string = .string + 1;
                    if string >= 'a .and string <= 'z
                       then char = string - $20;  /* CONVERT TO UPPER CASE */
                       else char = string;
                    if char
                       case 'N then crlf;
                       case 'B then putchar(bel);
                       case 'L then putchar(lf);
                       case 'R then putchar(cr);
                       case 'E then putchar(esc);
                       case '0 then putchar(nul);
                       else begin
                               putchar('\);
                               putchar(string);
                            end;
                 end;
            else putchar(string);
         .string = .string + 1;
      end;
endproc;


procedure space(integer n);
   while n > 0
      begin
         putchar(sp);
         n = n - 1;
      end;
endproc;

  task_util.lib

/* MULTITSK Utility Routines        Ver 1.00     7-Nov-86        fws */

/* These routine initialize, add, and delete tasks from the system.   */
/* The system IRQ routine is also in this file.                       */
```

```
   /*-------------------------------*/
   /*          add new task         */
   /*-------------------------------*/
   /* Add a new task to the system. No check is made to see if it is a  */
   /* valid task, so do it correctly!                                   */

   procedure add_task(byte task: integer .proc):
      integer .tmp_sp:

/* how many timer interrupts are allowed per unit of allicated time    */

      /* avoid swapping tasks at this time */
      swap_lock=1;

      /* get set up for the new task */
      new_task = task;  active_tasks = active_tasks + 1;
      task_stat(new_task) = ACTIVE;
      task_skips(new_task) = task_freq(new_task);
      .tmp_sp = stack_adr(new_task);
      tmp_sp = .proc;

      /* disable interrupts while manipulating the stacks */
      ccr = ccr or $10;
      old_stack = stack;

      stack = stack_adr(new_task);          /* set sp to new stack adr */
      gen $34,$7f;                       /* push info onto new stack */
      .stack_ccr = stack;                      /* point to new stack's ccr */
      stack_ccr = stack_ccr AND $ef;            /* enable irq in new stack ccr */
      stack_ptr(new_task) = stack;        /* store new sp into tsk sp array */

      stack = old_stack;
      ccr = ccr AND $EF;
      swap_lock = 0;

   endproc;

   /*-------------------------------*/
   /*          delete task          */
   /*-------------------------------*/
   /* Delete a task from the system. No check is made to see if it is a  */
   /* valid task, so do it correctly!                                    */

   procedure del_task(byte task);

      if active_tasks = 1     /* Must be at least one to be operational! */
         then return;

      /* avoid swapping tasks */
      swap_lock = 1;

      active_tasks = active_tasks - 1;
      task_stat(task) = INACTIVE;

      ccr = ccr OR $10;
      repeat    /* until you find the next active task */
         cur_task = cur_task + 1;
         if cur_task > LAST_TASK
            then cur_task = 0;

         if task_stat(cur_task) = ACTIVE
            then begin
                 if task_skips(cur_task) = 0
                    then break;
                 task_skips(cur_task) = task_skips(cur_task) - 1;
                 end;
      forever;

      task_skips(cur_task) = task_freq(cur_task);
      slice_cnt = SLICE_MAX;
      stack = stack_ptr(cur_task);              /* set stack to new task sp */
      ccr = ccr AND $EF;

      swap_lock = 0;

      gen $3b;   /* do a RTI, so the new stack will be "POPPED" */

   endproc;

   /*-------------------------------*/
   /*       interrupt routine       */
   /*-------------------------------*/


/* how many timer interrupts are allowed per unit of allicated time    */

     /* This is the system IRQ support routine. If other IRQ's need to be */
     /* supported, a call to a polling routine should be performed to     */
     /* determine the IRQ source, with this routine having the highest    */
     /* priority.                                                          */

     procedure IRQ;

        gen $10,$BE,MSB_YTMP,LSB_YTMP;          /* restore the Y reg..     */

        led0 = not(btmask(cur_task) OR $80);

        clear_clock_IRQ;

        /* check to see if we need to SWAP tasks */
        if slice_cnt > $00
           then slice_cnt = slice_cnt - 1;
```

```
    if active_tasks = $01 .OR swap_lock <> $00
       then begin
          led0 = not(btmsk(cur_task));
          return;
          end;

    if slice_cnt <= $00
       then begin

          /* save the current stack pointer */
          stack_ptr(cur_task) = stack;

          repeat   /* until you find the next active  task */

             cur_task = cur_task + 1;
             if cur_task > LAST_TASK
                then cur_task = 0;

             if task_stat(cur_task) = ACTIVE
                then begin
                   if task_skips(cur_task) = 0
                      then break;
                   task_skips(cur_task) = task_skips(cur_task) - 1;
                   end;
          forever;

          task_skips(cur_task) = task_freq(cur_task);
          slice_cnt = SLICE_MAX;
          num_swaps = num_swaps + 1;

          stack = stack_ptr(cur_task);   /* set stack to new task sp */
          end;

    led0 = not(btmsk(cur_task));

 endproc;

 /*-------------------------------*/
 /*  Multi-Tasker Initialisation */
 /*-------------------------------*/
 procedure init:byte cnt.value;

    clock_init;

    active_tasks = 0;   swap_lock = 0;   num_swaps = 0;

 /**************************************************************/
 /* This routine sets up the initial task priority levels. The      */
 /* value of task_freq(tasknumber) represents the time slice that  */
 /* the task software is actually executed. This unit of time is   */
 /*    SLICE_CNT * (IRQ RATE).(ie. task_freq=1 then time = 20ms)   */
    value=0;
    cnt=0;
    repeat
       task_freq(cnt)=value;


/* how many timer interrupts are allowed per unit of allicated time     */

       cnt=cnt+1;
       value=value+$20;
    until cnt=6;
 /**************************************************************/

 /* initialize individual task variables as necessary......         */
    key = $00;  keystat = $00;


    led0 = $ff;

    /* initialise task's stacks and status. Must do for each active */
    /* task in the system.                                          */
    stack_adr(0)=.stack0;   task_stat(0)=ACTIVE;     /* tasker     */
    stack_adr(1)=.stack1;   task_stat(1)=INACTIVE;   /* task_1     */
    stack_adr(2)=.stack2;   task_stat(2)=INACTIVE;   /* task_2     */
    stack_adr(3)=.stack3;   task_stat(3)=INACTIVE;   /* task_3     */
    stack_adr(4)=.stack4;   task_stat(4)=INACTIVE;   /* task_4     */
    stack_adr(5)=.stack5;   task_stat(5)=INACTIVE;   /* task_5     */

    /* intialize the "tasker" as the only active task */
    cur_task = 0;   active_tasks = 1;
    task_skips(0) = task_freq(0);   slice_cnt = SLICE_MAX;

 endproc;


  /* INTELLIGENT TERMINAL HANDLER LIBRARY   V:4.00  */


 include 0.iosubs.lib;


 /* THIS LIBRARY IS CONFIGURED FOR A SOROC IO120/LEAR SIEGLER ADM-5 */


 procedure nulls:byte count;
    count = 1;
    while count /* IMPLICIT <> 0 */
       begin
          putchar(null);
          count = count - 1;
       end;
 endproc;
```

```
          procedure erase_eol;
             putchar(esc);
             putchar('T);
             nulls;
          endproc;


          procedure erase_eop;
             putchar(esc);
             putchar('Y);
             nulls;
          endproc;


          procedure cursor(byte column.row);
             putchar(esc);
             putchar('=);
             putchar(sp + row);      /* OFFSET OF $20 */
             putchar(sp + column);   /* OFFSET OF $20 */
             nulls;
          endproc;


          procedure home;
             cursor(0,0);

/* how many timer interrupts are allowed per unit of allicated time     */
          endproc;


          procedure home_n_clr;
             home;
             erase_eop;
          endproc;


          procedure attr_on;
             putchar(esc);
             putchar(')');
          endproc;


          procedure attr_off;
             putchar(esc);
             putchar('(');
          endproc;


           /*   HEX-IO LIBRARY   V:4.00   */


          include 0.hexglobl.def;
          include 0.trufalse.def;
          include 0.iosubs.lib;


          procedure get_hex_nibble:byte inchar;
             inchar = get_uc;
             keychar = inchar;
             erflag = true;
             if inchar >=  '0 .and inchar <= '9
                then begin
                      inchar = inchar - '0;
                      erflag = false;
                      end;
                else if inchar >=  'A .and inchar <= 'F
                        then begin
                              inchar = inchar - '7;
                              erflag = false;
                              end;
          endproc inchar;


          procedure get_hex_byte:byte inchar;
                inchar = shift(get_hex_nibble,4);
                if erflag = true
                   then return;
                inchar = inchar or get_hex_nibble;
          endproc inchar;


          procedure get_hex_address:integer inchar;
                inchar = swap(integer(get_hex_byte));
                if erflag = true
                   then return;
                inchar = inchar or integer(get_hex_byte);
          endproc inchar;


          procedure put_hex_nibble(byte outchar);
             outchar = (outchar and $0f) + '0;  /* CONVERT TO ASCII */
             if outchar > '9
                then outchar = outchar + 7;        /* A-F OFFSET */
             putchar(outchar);
          endproc;


          procedure put_hex_byte(byte outchar);
             put_hex_nibble(shift(outchar,-4)); /* FIRST DIGIT */

/* how many timer interrupts are allowed per unit of allicated time     */
```

```
      put_hex_nibble(outchar);             /* LAST DIGIT */
   endproc;


   procedure put_hex_address(integer outchar);
      put_hex_byte(swap(outchar)); /* FIRST TWO DIGITS */
      put_hex_byte(byte(outchar)); /* LAST TWO DIGITS */
   endproc;
```

dumb_task.lib

```
   /* Dumb Task Examples            Ver 1.00        7-Nov-86       (ws  */

   /* The following routines are examples of how a task is added or    */
   /* deleted from the Multi-Tasker. As the digits 1-5 are typed, the  */
   /* corresponding task is alternately added or deleted, depending    */
   /* on its last status. Task_0 is used as a keyboard handler with    */
   /* the global variable of KEY, the key pressed, and KEYSTAT, an     */
   /* flag to indicate that a key has been pressed. This is also a     */
   /* good example of how to write tasks. The one constraint is that   */
   /* the task procedure names MUST NOT BE CHANGED, as they are used   */
   /* in other parts of the tasker to generate address pointers.       */
   /* The letters Q through T are used to modify the execution time    */
   /* of individual tasks. The P key is used to modify task0's         */
   /* execution time and the ESC key is used to return to FLEX. Be     */
   /* careful that you do not decrease task0's time to much as it will */
   /* result in a VERY SLOW scan for inputed cherators from the        */
   /* keyboard!!!!                                                     */

   procedure task_1: byte char;
      repeat
         dumb_cnt(1)=dumb_cnt(1)+1;
         if keystat <> 0 .AND key = '1 then
            begin
               keystat=0;
               del_task(1);
            end;
      forever;
   endproc;

   procedure task_2;
      repeat
         dumb_cnt(2)=dumb_cnt(2)+1;
         if keystat <> 0 .AND key = '2 then
            begin
               keystat=0;
               del_task(2);
            end;
      forever;
   endproc;

   procedure task_3;
      repeat
         dumb_cnt(3)=dumb_cnt(3)+1;
         if keystat <> 0 .AND key = '3 then
            begin
               keystat=0;
               del_task(3);
            end;
      forever;
   endproc;

   procedure task_4;
      repeat
         dumb_cnt(4)=dumb_cnt(4)+1;
         if keystat <> 0 .AND key = '4 then
            begin
               keystat=0;
               del_task(4);
            end;
      forever;
   endproc;

/* how many timer interrupts are allowed per unit of allicated time    */
   procedure task_5;
      repeat
         dumb_cnt(5)=dumb_cnt(5)+1;
         if keystat <> 0 .AND key = '5 then
            begin
               keystat=0;
               del_task(5);
            end;
      forever;
   endproc;

   procedure clear_status_line;
      cursor(0,23);
      erase_eop;
   endproc;

   procedure wait(byte cnt):integer i;
      while cnt
         begin
            cnt=cnt-1;
            i=$4000;
            while i i=i-1;
         end;
   endproc;

   /* This is task_0. It is usally used for system Tasker functions.   */
   /* It should never be disabled. In this example of its use, the     */
   /* keyboard is checked for input. Key is used to store the value,   */
   /* while keystat is used to indicate a valid key to all other tasks. */
   /* The value of key is then used to enable other tasks if they are  */
   /* currently inactive. See task_1 thru task_5 for other interaction. */
```

```
   procedure task_0: byte cnt;

      cnt=0;
      term_stat=0;
      repeat
         dumb_cnt(cnt)=0;
         cnt=cnt+1
      until cnt > LAST_TASK;


      home_n_clr;                                   /* Home & clear screen */

      print(.copyright); crlf;
      space(9); print(.notice); crlf;
      space(7); print(.author);
      cursor(0,5);
      print("                          Skip        task\n");
      print("         Status           count      counter\n");

      repeat    /* FOREVER */
      dumb_cnt(0)=dumb_cnt(0)+1;

         /* Conditionally add tasks.    */
      if keystat = 0 then
         begin
            key=getkey;
            if key <> nul then keystat = 1;
            if key > $60 then key=key-$20  /* Make Upper case   */
            if key
               case '0 then
                  begin
                     cursor(0,23);
                     print("You can't delete this task!!!!");
                     keystat=0;
                     wait(10);
                     clear_status_line;
                  end;
               case '1 then
                  begin
/* how many timer interrupts are allowed per unit of allicated time    */
                     if task_stat(1)=INACTIVE then
                        begin
                           add_task(1,.task_1);
                           keystat=0;
                        end;
                     end;
               case '2 then
                  begin
                     if task_stat(2)=INACTIVE then
                        begin
                           add_task(2,.task_2);
                           keystat=0;
                        end;
                     end;
               case '3 then
                  begin
                     if task_stat(3)=INACTIVE then
                        begin
                           add_task(3,.task_3);
                           keystat=0;
                        end;
                     end;
               case '4 then
                  begin
                     if task_stat(4)=INACTIVE then
                        begin
                           add_task(4,.task_4);
                           keystat=0;
                        end;
                     end;
               case '5 then
                  begin
                     if task_stat(5)=INACTIVE then
                        begin
                           add_task(5,.task_5);
                           keystat=0;
                        end;
                     end;
               case 'Q then
                  begin
                     cursor(0,23);
                     print("Task1 new priority (00-ff): ");
                     task_freq(1)=get_hex_byte;
                     clear_status_line;
                     keystat=0;
                  end;
               case 'W then
                  begin
                     cursor(0,23);
                     print("Task2 new priority (00-ff): ");
                     task_freq(2)=get_hex_byte;
                     clear_status_line;
                     keystat=0;
                  end;
               case 'E then
                  begin
                     cursor(0,23);
                     print("Task3 new priority (00-ff): ");
                     task_freq(3)=get_hex_byte;
                     clear_status_line;
                     keystat=0;
                  end;
               case 'R then
                  begin
```

```
                        cursor(0,23);
                        print("Task4 new priority (00-ff): ");
                        task_freq(4)=get_hex_byte;
                        clear_status_line;
                        keystat=0;
                     end;
                  case 'T then
                     begin


/* how many timer interrupts are allowed per unit of allicated time     */

                        cursor(0,23);
                        print("Task5 new priority (00-ff): ");
                        task_freq(5)=get_hex_byte;
                        clear_status_line;
                        keystat=0;
                     end;
                  case 'P then
                     begin
                        cursor(0,23);
                        print("Task0 new priority (00-ff): ");
                        task_freq(0)=get_hex_byte;
                        clear_status_line;
                        keystat=0;
                     end;
                  case ESC then               /* Exit to flex....    */
                     begin
                        ccr=ccr or $10;
                        clk_irq_off;
                        led0=$ff;
                        home_n_clr;
                        warms;
                     end;
               else keystat = 0;    /* none of the above, so ignore.   */
            end;

      /* Display the task data.   */
            cnt=0;
            cursor(0,7);
            repeat
               print("Task"); putchar(cnt+$30);space(3);
               if task_stat(cnt) = ACTIVE then print("ACTIVE   ");
                  else print("INACTIVE ");
               space(7);
               put_hex_byte(task_freq(cnt));
               space(10); put_hex_byte(dumb_cnt(cnt));
               cnt=cnt+1;
               crlf;
            until cnt > LAST_TASK;
            cursor(0,7);

         forever;
      endproc;
```

multitsk.pl9

```
    /* Multi-Tasking Kernal - Main Routines     Ver 1.0  (7-November-86 fws/*/

    /*                     DISTRIBUTION RIGHTS

       I allow unrestricted distribution of this software on a non-profit
    basis.  However, any commercial application using either the programs
    or documentation requires my written permission in advance.
                                                                        */




    /* The following    statement defines a temp location for Reg "Y"   */
    /* and is the only fixed RAM variable used in this software. It     */
    /* will usually be just above the stack pointer. Saving and         */
    /* restoring this register is done with GEN statements in the       */
    /* TASKER and IRQ procedures.                                        */

    constant MSB_YTMP=$BF, LSB_YTMP=$FE;      /* Temp Yreg Address       */

    /* This program also uses stack memory for each task. Be sure you   */
    /* allow enough memory for the stack area.                           */

    origin=$B000;  stack=$AFFE;               /* Defined by user.    */

    /*-------------------------------*/


/* how many timer interrupts are allowed per unit of allicated time     */
     /*       required libraries       */
     /*--------------------------------*/
     Include 1.global.lib;              /* MULTITSK globals                   */
     include 1.iosubs.lib;              /* Use your own routines if different */
     include 1.hexio.lib;               /* Ditto !                            */
     include 1.termsubs.lib;            /* Ditto !                            */
     include 1.clock.li ;               /* MULTITSK interrupt source          */
     include 1.tsk_utl.lib;             /* MULTITSK utilities                 */
     include 1.dumb_tsk.lib;            /* MULTITSK example                   */

     /*--------------------------------*/
     /*        task controller         */
     /*--------------------------------*/
     procedure tasker: byte cnt;

        gen $10,$BF,MSB_YTMP,LSB_YTMP;   /* Save the Y reg in Y_TEMP     */

        /* disable system interrupts and set "E" flag */
        ccr=ccr or $90;
```

```
       init;           /* task_0 set ACTIVE in init...      */

       /* initialize tasks that will be active when you start the program */
    /* Also set the number of active tasks in the init procedure          */
    /*
       add_task(1,.task_1);      (Only task_0 is active in this example)
       add_task(2,.task_2);
       add_task(3,.task_3);
       add_task(4,.task_4);
       add_task(5,.task_5);
    */

       /* reset the stack, enable interrupts, and GO */
       stack = .stack0 + 1;
       clk_irq_on;
       ccr = ccr AND $EF;    /* enables system interrupts */

       repeat
          task_0;
       forever;

    endproc;
```

tasker.lis

```
    2.multitsk.pl9
    2.global.lib;
    2.clock.lib;
    2.tsk_utl.lib;
    2.dumb_tsk.lib;
```

EOF

Simple Winchester

by: Samuel I. Green Ph.D.

Continued From Last Month

```
        ************************************************************
        * HARD DISK FORMAT ROUTINE.  WRITTEN BY R. LEFF 5/26/82 AND   *
        * MODIFIED BY D.J. GRAVES 7/12/82 WITH SELF CONTAINED DRIVERS *
        * AND ADDED COMMENTS.  VALID FOR WESTERN DIGITAL WD1000 CON-  *
        * TROLLER AND FLEX9.  PROGRAM FORMATS, LINKS, REPORTS ERRORS, & *
        * SETS UP DIRECTORY AND SYSTEM INFO SECTORS.  IT DOES NOT MAP *
        * OUT BAD SECTORS.  from 68 uJournal  Nov '82   pg 25         *
        * modified for true data bus and looping read/write macros    *
        * and the WD1000-05 Winchester Disk Controller Board          *
        * by Samuel I. Green, 13052 Ferntrails, Creve Coeur, Mo 63141 *
        ************************************************************

 CD03   WARMS   EQU    $CD03
 CD39   OUTDEC  EQU    $CD39
 CD15   GETCHR  EQU    $CD15
 CD1B   INBUFF  EQU    $CD1B
 CD33   SETEXT  EQU    $CD33
 D406   FMS     EQU    $D406
 CD2D   GETFIL  EQU    $CD2D
 CD18   PUTCHR  EQU    $CD18
 CD0F   OUTCH   EQU    $CD0F
 CD1E   PSTRNG  EQU    $CD1E
 CD24   PCRLF   EQU    $CD24
 CD48   INDEC   EQU    $CD48
 CD42   GETHEX  EQU    $CD42
 CD45   OUTADR  EQU    $CD45
 CD4E   STAT    EQU    $CD4E
 CD3F   RPTERR  EQU    $CD3F
 CD3C   OUTHEX  EQU    $CD3C
 CD27   NXTCH   EQU    $CD27
 CC0E   DATE    EQU    $CC0E


 E030   DBASE   EQU    $E030     SYSTEM DEPENDENT: FIRST ADDRESS OF DISK PORT
 E030   RDATA   EQU    DBASE     REGISTER 1 FOR READ PURPOSES
 E030   WDATA   EQU    DBASE     REGISTER 1 FOR WRITE PURPOSES
 E031   ERROR   EQU    DBASE+1   REGISTER 2, ETC. FOLLOW IN ORDER
 E031   WPCR    EQU    DBASE+1   SEE WD1000 MANUAL FOR DEFINITION OF PURPOSES
 E032   RSCNT   EQU    DBASE+2
 E032   WSCNT   EQU    DBASE+2
 E033   RSECNO  EQU    DBASE+3
 E033   WSECNO  EQU    DBASE+3
 E034   RCYLL   EQU    DBASE+4
 E034   WCYLL   EQU    DBASE+4
 E035   RCYLH   EQU    DBASE+5
 E035   WCYLH   EQU    DBASE+5
 E036   RSDH    EQU    DBASE+6
 E036   WSDH    EQU    DBASE+6
 E037   STATUS  EQU    DBASE+7
 E037   COMAND  EQU    DBASE+7

 0020   SECMAX  EQU    32


                  00FF   TRKMAX   EQU    255
                  0050   FORMAT   EQU    $50        uninverted for true data bus
                  0020   TRKCNT   EQU    32
                  0014   RESTAL   EQU    $14
                  0010   REST     EQU    $10


 0000                             ORG    0
 0000 20    01           START    BRA    GO
 0002 03                          FCB    3          VERSION 3
```

```
                         * INITIALIZE THE DRIVE

0003 86     00           GO        LDA    #$00
0005 B7     E036                   STA    WSDH      CHOOSE DRIVE,SIDE,HEAD AS ZERO
0008 86     20                     LDA    #PRECMP
000A B7     E031                   STA    WPRE      SET PRECOMPENSATION TRACK
000D 86     00                     LDA    #00
000F B7     E035                   STA    WCYLH     CYLINDER HIGH IS ALWAYS ZERO ON 5 MEG DRIVE
0012 86     10                     LDA    #REST
0014 17     044E                   LBSR   COMOUT    NOW DO A  STORE AND SET SEEK SPEED
0017 C6     00                     LDB    #0

                         * MAIN FORMATTING PROGRAM BEGINS HERE

0019 8E     0265                   LDX    #GOMESS   READY TO FORMAT? Y/N
001C BD     CD1E                   JSR    PSTRNG
001F BD     CD15                   JSR    GETCHR
0022 81     59                     CMPA   #'Y
0024 1026   CCE0                   LBNE   WARMS
0028 8E     0247         ASKD      LDX    #DRVMES   WHICH DRIVE? 2/3
002B BD     CD1E                   JSR    PSTRNG
002E BD     CD15                   JSR    GETCHR
0031 80     30                     SUBA   #'0
0033 7F     02BD                   CLR    SIDENO
0036 B7     02BE                   STA    SIDENO+1
0039 81     03                     CMPA    3        SEE IF VALID DRIVE NO.
003B 22     EB                     BHI    ASKD
003D 81     01                     CMPA   #1
003F 23     E7                     BLS    ASKD
0041 BD     040F                   JSR    SELECT    SELECT THE DRIVE
0044 8E     0288                   LDX    #TRKMES   HOW MANY TRACKS? (1 TO 256 PERMITTED)
0047 BD     CD1E                   JSR    PSTRNG
004A BD     CD1B                   JSR    INBUFF    GET DECIMAL NUMBER
004D BD     CD48                   JSR    INDEC     AND CONVERT TO BINARY
0050 30     1F                     LEAX   -1,X      (TRACKS ACTUALLY RUN 0 TO 255)
0052 BF     02C0                   STX    TRKCNT
0055 8E     026C                   LDX    #SECMES   HOW MANY SECTORS? (1-32)
0058 BD     CD1E                   JSR    PSTRNG
005B BD     CD1B                   JSR    INBUFF    ETC. AS ABOVE
005E BD     CD48                   JSR    INDEC
0061 1F     10                     TFR    X,D       PUT SECTOR NUMBER IN D
0063 F7     02BF                   STB    SECCNT    AND SAVE L.S. BYTE
0066 BD     03C6                   JSR    ILVE       CALCULATE INTERLEAVE TABLE
0069 4F                            CLRA             START WITH TRACK ZERO
006A 34     02           FALL      PSHS   A
006C BD     01B1                   JSR    TFORMT    FORMAT A TRACK
006F 35     02                     PULS   A
0071 B1     02C1                   CMPA   TRKCNT+1
0074 27     03                     BEQ    DOLINK
0076 4C                            INCA
0077 20     F1                     BRA    FALL
0079 7F     0205         DOLINK    CLR    EOTFLG    CLEAR END OF TRACK FLAG
007C B6     02C1                   LDA    TRKCNT+1  GET TRACK
007F F6     02BF                   LDB    SECCNT
0082 FD     02C2                   STD    LNKEND    SAVE END OF SECTOR CHAIN
0085 8E     02C8                   LDX    #WORK+2   CLEAR THE BUFFER
0088 6F     80           DOLIN1    CLR    ,X+
008A 8C     03C8                   CMPX   #WORK+256
008D 26     F9                     BNE    DOLIN1
008F 7F     0221                   CLR    DTEMP     START AT TRACK ZERO
0092 7F     0220         DOLIN2    CLR    SECTOR    FIRST SECTOR
0095 7C     0220         DOLIN3    INC    SECTOR
0098 F6     0220                   LDB    SECTOR    GET SECTOR
009B F7     0222                   STB    DTEMP+1   SAVE SECTOR
009E B6     0221                   LDA    DTEMP     GET TRACK
00A1 10B3   02C2                   CMPD   LNKEND    SEE IF LAST ONE
00A5 26     08                     BNE    DOLIN5
00A7 73     02C5                   COM    EOTFLG    ALSO END OF TRACK
00AA CC     0000                   LDD    #0        END OF CHAIN
00AD 20     0B                     BRA    LINKS3
00AF F1     02BF         DOLIN5    CMPB   SECCNT    SEE IF LINK SHOULD BE TO NEXT TRACK
00B2 26     05                     BNE    LINKS1
00B4 73     02C5                   COM    EOTFLG
00B7 5F                            CLRB             IT IS; LINK TO NEXT TRACK
00B8 4C                            INCA             INCREMENT LINK (NOT TRACK)
00B9 5C                  LINKS1    INCB
00BA 8E     02C6         LINKS3    LDX    #WORK
00BD ED     84                     STD    ,X
00BF FC     0221                   LDD    DTEMP
00C2 BD     04B9                   JSR    WRITEC    WRITE IT OUT; IGNORE ERRORS
00C5 27     17                     BEQ    LINKS9
00C7 FC     0221         OFERR    LDD    DTEMP     on write error reformat track
00CA BD     01B1                   JSR    TFORMT
00CD 8E     0224                   LDX    #ERRMES   OUTPUT AN ERROR MESSAGE
00D0 BD     CD1E                   JSR    PSTRNG
00D3 B6     0221                   LDA    DTEMP
00D6 BD     01F4                   JSR    HEXOUT
00D9 B6     0221                   LDA    DTEMP
00DC 0E     92                     JMP    DOLIN2    write sector to reformatted track
00DE 7D     02C5         LINKS9    TST    EOTFLG    END OF TRACK?
00E1 27     B2                     BEQ    DOLIN3
00E3 7F     02C5                   CLR    EOTFLG
00E6 F6     0221                   LDB    DTEMP     SEE IF LAST TRACK
00E9 F1     02C1                   CMPB   TRKCNT+1
00EC 27     05                     BEQ    SIDNC
00EE 7C     0221                   INC    DTEMP     INCREMENT TRACK

00F1 20     9F                     BRA    DOLIN2

                         * SET UP THE S.I.R.

00F3 8E     02C6         SIDNC     LDX    #WORK
00F6 6F     80           SIP       CLR    ,X+
00F8 8C     02D6                   CMPX   #WORK+16
00FB 26     F9                     BNE    SIP
00FD 108E   023C                   LDY    #DISNAM   WRITE OUT "WINCHESTER" AS DISK NAME
0101 A6     A0           NONAM     LDA    ,Y+
0103 A7     80                     STA    ,X+
0105 8C     02E1                   CMPX   #WORK+27
0108 26     F7                     BNE    NONAM
010A FC     02BD                   LDD    SIDENO    PUT ON THE DISK SIDE NUMBER
010D ED     81                     STD    X++       (AS VOLUME NUMBER)
010F CC     0101                   LDD    #0101     FREE CHAIN STARTS HERE (PREVIOUS SECTORS DIR)
0112 ED     81                     STD    X++
0114 FC     02C2                   LDD    LNKEND    END OF FREE CHAIN
0117 ED     81                     STD    X++
0119 B6     02C1                   LDA    TRKCNT+1
011C F6     02BF                   LDB    SECCNT
011F 3D                            MUL
0120 ED     81                     STD    X++
0122 FC     CC0E                   LDD    DATE      MONTH,DAY
0125 ED     81                     STD    X++
0127 B6     CC10                   LDA    DATE+2    YEAR
012A A7     80                     STA    X+
012C FC     02C2                   LDD    LNKEND    TRACK AND SECTOR MAXIMUM
012F ED     81                     STD    X++
0131 6F     80           CLRST     CLR    X+
0133 8C     03C6                   CMPX   #WORK+256
0136 26     F9                     BNE    CLRST
0138 CC     0003                   LDD    #0003     POINT TO TRACK 0, SECTOR 3
013B 8E     02C6                   LDX    #WORK
013E 34     06                     PSHS   D
0140 BD     04B9                   JSR    WRITEC    WRITE OUT THE SECTOR
0143 35     06                     PULS   D
0145 8E     02C6                   LDX    #WORK
0148 BD     0477                   JSR    READC     AND CHECK IT
014B 26     22                     BNE    SIFERR
014D 8E     02C6                   LDX    #WORK     FIX LINK FOR END OF DIR. FREE CHAIN
0150 6F     80           SIP6      CLR    X+
0152 8C     03C6                   CMPX   #WORK+256
0155 26     F9                     BNE    SIP6
0157 8E     02C6                   LDX    #WORK
015A 4F                            CLRA
015B F6     02BF                   LDB    SECCNT    LAST SECTOR OF DIRECTORY
015E 34     06                     PSHS   D
0160 BD     04B9                   JSR    WRITEC
0163 35     06                     PULS   D
0165 8E     02C6                   LDX    #WORK
0168 BD     0477                   JSR    READC
016B 1027   CD94                   LBEQ   WARMS

016F 8E     0178         SIFERR    LDX    #SIFMES
0172 BD     CD1E                   JSR    PSTRNG
0175 7E     CD03                   JMP    WARMS
0178 0A0D                SIFMES    FDB    $A0D
017A 57 52 49 54                   FCC    'WRITE ERROR ON SYSTEM INFORMATION SECTOR OR DIRECTORY!',4
017E 45 20 45 52
0182 52 4F 52 20
0186 4F 4E 20 53
018A 59 53 54 45
018E 4D 20 49 4E
0192 46 4F 52 4D
0196 41 54 49 4F
019A 4E 20 53 45
019E 43 54 4F 52
01A2 20 4F 52 20
01A6 44 49 52 45
01AA 43 54 4F 52
01AE 59 21 04

                         * HERE WE DO THE FORMATTING
                         * 256 BYTES/SECTOR, MAX. 32 SECTORS/TRACK

01B1 F6     02BF         TFORMT    LDB    SECCNT    SECTOR COUNT MUST BE RELOADED EACH TIME
01B4 F7     E032                   STB    WSCNT
01B7 C6     0C                     LDB    #$0C      LOAD GAP INFO
01B9 F7     E033                   STB    WSNUMB
01BC 12                            NOP
01BD 12                            NOP
01BE B7     E034                   STA    WCYLL     output the track
01C1 86     50                     LDA    #FORMAT   SEND "FORMAT TRACK" COMMAND
01C3 17     029F                   LBSR   COMOUT
01C6 B6     E037         FORMD     LDA    STATUS
01C9 85     08                     BITA   #$08      DATA REQUEST?
01CB 27     F9                     BEQ    FORMD
01CD 8E     0461                   LDX    #SECTAB   POINT TO SKEWTABLE
01D0 5F                            CLRB
01D1 86     00           MFORMA    LDA    #00       NO BAD BLOCKS MARKED ON DISK
01D3 B7     E030                   STA    WDATA
01D6 A6     80                     LDA    0,X+
01D8 B7     E030                   STA    WDATA     WRITE THE SECTOR NUMBER
01DB CB     02                     ADDB   #2        NOTE THAT 256 BYTES MUST BE TRANSMITTED EVEN THOUGH
01DD 26     F2                     BNE    MFORMA    ONLY 2X # OF SECTORS IS REAL DATA (REST IS GARBAGE)
01DF B6     E037         FORM2     LDA    STATUS
01E2 85     80                     BITA   #$80      BUSY?
01E4 26     F9                     BNE    FORM2     WAIT FOR COMMAND TO FINISH
01E6 85     01                     BITA   #$01        BUSY
01E8 26     01                     BNE    FERR
01EA 39                            RTS

01EB 8E     020B         FERR      LDX    #FMESS
01EE BD     CD1E                   JSR    PSTRNG
01F1 B6     E031                   LDA    ERROR
01F4 34     02           HEXOUT    PSHS   A         SAVE THE CHARACTER
01F6 44                            LSRA
01F7 44                            LSRA
01F8 44                            LSRA
01F9 44                            LSRA
01FA 8D     04                     BSR    OUT2
01FC 35     02                     PULS   A
01FE 84     0F                     ANDA   #$F
0200 81     09           OUT2      CMPA   #9
0202 23     02                     BLS    OUT1
0204 8B     07                     ADDA   #7
0206 8B     30           OUT1      ADDA   #$30
0208 7E     CD0F                   JMP    OUTCH

020B 46 4F 52 4D         FMESS     FCC    'FORMAT TRACK ERROR: ',4
020F 41 54 20 54
0213 52 41 43 4B
0217 20 45 52 52
021B 4F 52 3A 20
021F 04
0220                     SECTOR    RMB    1
0221                     DTEMP     RMB    2
0223                     FLAG      RMB    1
0224                     CHNSTD    RMB    2
0226                     CHNEND    RMB    2
0228                     CHNCNT    RMB    2
022A 52 65 66 6F         ERRMES    FCC    'Reformat Track # ',4
022E 72 6D 61 74
0232 20 54 72 61
0236 63 6B 20 23
023A 20 04
023C 57 49 4E 43         DISNAM    FCC    'WINCHESTER '
0240 48 45 53 54
0244 45 52 20
0247 57 48 49 43         DRVMES    FCC    'WHICH DRIVE? (2 OR 3) ',4
024B 48 20 44 52
024F 49 56 45 3F
0253 20 28 32 20
0257 4F 52 20 33
025B 29 20 04
025E 46 4F 52 4D         GOMESS    FCC    'FORMAT DRIVE ',4
0262 41 54 20 44
0266 52 49 56 45
026A 20 04
```

```
026C 0A0D          SECMES  FDB     $0A0D
026E 48 4F 57 20           FCC     'HOW MANY SECTORS? (1-32) ',4
0272 4D 41 4E 59
0276 20 53 45 43
027A 54 4F 52 53
027E 3F 20 28 31
0282 2D 33 32 29
0286 20 04
0288 0A0D          TRKMES  FDB     $0A0D
028A 48 4F 57 20           FCC     'HOW MANY TRACKS? (1-256)',4
028E 4D 41 4E 59
0292 20 54 52 41
0296 43 4B 53 3F
029A 20 28 31 2D
029E 32 35 36 29
02A2 04
02A3 0A0D          OKMES   FDB     $0A0D
02A5 52 45 41 44           FCC     'READY TO FORMAT? (Y/N) ',4
02A9 59 20 54 4F
02AD 20 46 4F 52
02B1 4D 41 54 3F
02B5 20 28 59 2F
02B9 4E 29 20 04
02BD               SIDENO  RMB     2
02BF               SECCNT  RMB     1
02C0               TRKCNT  RMB     2
02C2               LNKEND  RMB     2
02C4               SIDE    RMB     1
02C5               SETFLG  RMB     1
02C6               WORK    RMB     256

                   * CALCULATE INTERLEAVE TABLE

03C6 F6    02BF    INLVE   LDB     SECCNT
03C9 4F                    CLRA
03CA C3    04E1            ADDD    #SECTAB
03CD FD    044D            STD     TABEND
03D0 8E    042E            LDX     #INTMES   GET THE INTERLEAVE TABLE
03D3 BD    CD1E            JSR     PSTRNG
03D6 BD    CD1B            JSR     INBUFF
03D9 BD    CD48            JSR     INDEC
03DC 1F    10              TFR     X,D
03DE F7    044B            STB     INTER
03E1 86    FF              LDA     #-1       INITIALIZE ALL SECTORS AS UNUSED
03E3 8E    04E1            LDX     #SECTAB
03E6 A7    80      SETFLO  STA     ,X+
03E8 BC    044D            CMPX    TABEND
03EB 26    F9              BNE     SETFLO
03ED 7F    044A            CLR     COUNTR
03F0 7F    044C            CLR     RES
03F3 8E    04E1            LDX     #SECTAB
03F6 B6    044C    SETUP   LDA     RES
03F9 B1    02BF            CMPA    SECCNT
03FC 25    06              BLO     SETUP1
03FE B0    02BF            SUBA    SECCNT
0401 B7    044C            STA     RES
0404 F6    044C    SETUP1  LDB     RES
0407 A6    85              LDA     B,X
0409 81    FF              CMPA    #-1
040B 26    07              BNE     ELSE
040D B6    044A            LDA     COUNTR
0410 A7    85              STA     B,X
0412 20    05              BRA     OTHER

0414 7C    044C    ELSE    INC     RES
0417 20    DD              BRA     SETUP
0419 B6    044C    OTHER   LDA     RES
041C BB    044B            ADDA    INTER
041F B7    044C            STA     RES
0422 7C    044A            INC     COUNTR
0425 B6    02BF            LDA     SECCNT
0428 B1    044A            CMPA    COUNTR
042B 26    C9              BNE     SETUP
042D 39                    RTS

042E 0A0D          INTMES  FDB     $0A0D
0430 49 4E 50 55           FCC     'INPUT INTERLEAVE FACTOR: ',4
0434 54 20 49 4E
0438 54 45 52 4C
043C 45 41 56 45
0440 20 46 41 43
0444 54 4F 52 3A
0448 20 04
044A               COUNTR  RMB     1
044B               INTER   RMB     1
044C               RES     RMB     1
044D               TABEND  RMB     2

                   * DRIVE SELECT ROUTINE

044F 81    03      SELECT  CMPA    #3
0451 27    09              BEQ     ONE
0453 86    00              LDA     #$00
0455 B7    E036            STA     WSDH
0458 5F                    CLRB
0459 1C    FE              CLC
045B 39                    RTS
045C 86    01      ONE     LDA     #$01
045E B7    E036            STA     WSDH
0461 5F                    CLRB
0462 1C    FE              CLC
0464 39                    RTS


                   * SEND COMMAND TO WD1005

0465 34    04      COMOUT  PSHS    B
0467 F6    E037    WAIT    LDB     STATUS
046A C5    80              BITB    #$80      CHECK FOR BUSY
046C 26    F9              BNE     WAIT      *IF BUSY, WAIT FOR IT
046E C5    02              BITB    #$02      CHECK FOR COMMAND IN PROCESS
0470 26    F5              BNE     WAIT      WAIT FOR CIP TO END
0472 B7    E037            STA     COMAND    ISSUE COMMAND
0475 35    84      CONQ    PULS    B,PC
           00E0            SETDP   DBASE/256 THIS IS NEEDED TO SUPPRESS WARNINGS AT ASSEMBLY

                    * READ A SECTOR
                    * A HAS TRACK, B HAS SECTOR, X HAS BUFFER ADDRESS

           0020    READ    EQU     $20
           0030    WRITE   EQU     $30
```

```
0477 34    08      READC   PSHS    DP        SAVE DIRECT PAGE
0479 34    02              PSHS    A         SAVE TRACK TEMPORARILY
047B 5A                    DECB              FIX SECTOR FOR FLEX
047C 86    E0              LDA     #DBASE/256 GET DIRECT PAGE OF DISK INTERFACE
047E 1F    8B              TFR     A,DP
0480 35    02              PULS    A         RECOVER TRACK
0482 D7    33              STB     <RSECNO   GIVE IT THE SECTOR
0484 12                    NOP
0485 12                    NOP
0486 97    34              STA     <RCYLL    AND THE TRACK
0488 86    20              LDA     #READ     ISSUE A "READ SECTOR" COMMAND
>048A 17   FFD8            LBSR    COMOUT
048D 96    37      READ2   LDA     <STATUS
048F 85    80              BITA    #$80      BUSY?
0491 26    FA              BNE     READ2     WAIT FOR COMMAND TO COMPLETE
0493 96    37      READ3   LDA     <STATUS
0495 85    08              BITA    #$08      DATA REQUEST?
0497 27    FA              BEQ     READ3     WAIT FOR DATA REQUEST
           0499    NREAD   EQU     *
0499 5F                    CLRB
049A 96    30      LOOP    LDA     <RDATA
049C A7    80              STA     0,X+
049E 5A                    DECB
049F 26    F9              BNE     LOOP
04A1 D6    37              LDB     <STATUS
04A3 C5    01              BITB    #1        ERROR?
04A5 26    04              BNE     EREAD
04A7 C6    00              LDB     #0
04A9 35    88              PULS    DP,PC     RETURN WITH NO ERRORS FOUND
04AB 96    31      EREAD   LDA     <ERROR    ERROR, CHECK IT
04AD 81    40              CMPA    #%01000000 DATA CRC ERROR?
04AF 26    04              BNE     READ6
04B1 C6    08              LDB     #8        MUST BE CRC
04B3 35    88              PULS    DP,PC
04B5 C6    10      READ6   LDB     #$10      ID ERROR FOR FLEX
04B7 35    88              PULS    DP,PC


                   * ROUTINE TO WRITE OUT A SECTOR TO DISK

04B9 30    0A      WRITEC  PSHS    DP,A      THIS WORKS JUST ABOUT LIKE "READC"
04BB 5A                    DECB
04BC 86    E0              LDA     #DBASE/256
04BE 1F    8B              TFR     A,DP
04C0 35    02              PULS    A
04C2 D7    33              STB     <WSECNO
04C4 12                    NOP

04C5 12                    NOP
04C6 97    34              STA     <WCYLL
04C8 86    30              LDA     #WRITE
>04CA 17   FF98            LBSR    COMOUT
04CD A6    80      PUTLP   LDA     0,X+
04CF D6    37      PUTLP1  LDB     <STATUS
04D1 C5    08              BITB    #$08      DRQ?
04D3 26    08              BNE     WRTBYT
04D5 C5    80              BITB    #$80      BUSY?
04D7 26    F6              BNE     PUTLP1
04D9 C5    01              BITB    #$01      ERR?
04DB 35    88              PULS    DP,PC
04DD 97    30      WRTBYT  STA     <WDATA
04DF 20    EC              BRA     PUTLP
           04E1    SECTAB  EQU     *
                           END     START

ERROR(S) DETECTED

SYMBOL TABLE:

COMBUF 0220  CHKEND 0226  CHKSTR 022A  CLASPT 0131  CONQ   0475
COMAND E037  COMOUT 0465  COUNTR 044A  DATE   000E  DISRAM 023C
DMESS  025E  DOLIN1 0088  DOLIN2 0092  DOLIN3 0095  DOLIN5 00AF
DOLINX 0079  ENTRES 0267  DTEMP  0221  ELSE   0414  SETFLG 02C5
ADRMES 022A  ERROR  E031  FAIL   006A  FERR   01EB  FLAG   0223
FMES3  020B  FMS    D400  FORMD  01C6  FORM2  01DF  FORMAT 0050
GETCHR CD15  GETFIL CD2D  GETHEX CD42  GO     0003  GOMES3 0243
OTHER  0419  DBASE  E030  HEXOUT 01F4  INBUFF CD1B  INDEC  CD48
INLVE  03C6  INTER  044B  INTMES 042E  LOAD   000E  LINES1 0089
LINES3 00BA  LNKEND 02C2  LOOP   049A  NFORMA 0101  MORRAM 0101
NREAD  0499  NDRV   0028  SETCH  CD27  ONE    045C  ONTERR 00C7
OUT1   0206  OUT2   0300  OUTADD CD45  OUTCH  CD0F  OUTDEC CD39
OUTHEX CD3C  PCRLF  CD24  PRECMP 0020  PSTRNG CD1E  PUTCHR CD18
PUTLP  04CD  PUTLP1 04CF  RCYLH  E035  RCYLL  E034  RDATA  E030
READ   0020  READ2  048D  READ3  0493  READ6  04B5  EREAD  04AB
READC  0477  RES    044C  ROT    0010  RESTRL 0014  RPTERR CD3F
RSCNT  E032  RSDH   E036  RSECNO E033  SECCNT 02BF  SECMAX 0020
SECMES 026C  SECTAB 04E1  SECTOR 0220  SELECT 044F  SETEXT CD33
SETFLO 03E6  SETUP  03F6  SETUP1 0404  SIDE   02C4  SIDENO 02BD
SKIP   00FA  SKIPS  0150  SKPERR 016F  SKPERS 0178  SIZEC  00F3
START  0000  STAT   CD4E  STATUS E037  TABEND 044D  TFORMC 0181
TRKCNT 02C0  TEMPAD 00FF  TRKMES 0288  WAIT   0467  WARMS  CD03
WCYLH  E035  WCYLL  E034  WDATA  E030  WORK    2C6  WPRE   E031
WRITE  0030  WRITEC 04B9  WRTBYT 04DD  WSCNT  E032  WSDH   E036
WSECNO E033
```

```
              ******************************************************
              *        PARK.CMD for WDDRIVE and TDW                *
              * safely positions head at track above highest       *
              * used track number and exits Flex to monitor        *
              * by  Samuel E. Green, 13057 [illegible] Ln.         *
              *     [illegible], MD [illegible]      Jan '86       *
              ******************************************************

           E030  BASE    EQU     $E030     base address of host adapter
           FF00  MONITR  EQU     $FF00

C100                     ORG     $C100
C100 8E    E030  START   LDX     #BASE
C103 86    01            LDA     #1        track 256 plus 44 = 300
C105 A7    05            STA     5,X       cylinder high
C107 86    2C            LDA     #44
C109 A7    04            STA     4,X       cylinder low
C10B 86    70            LDA     #$70      seek
C10D A7    07            STA     7,X       command register
C10F 7E    FF00          JMP     MONITR
                         END     START

0 ERROR(S) DETECTED

SYMBOL TABLE:

BASE   E030   MONITR FF00   START  C100
```

EOF

Bradford Taylor
Sharm Engineering
Box 97
Mulvane, Ks 67110
Tel. (316) 777-0706

68' Micro Journal
5900 Cassandra Smith Rd.
Hixson TN 37343

Dear Don,

After reading the August 86 Mickey Ferguson review of the HIER software package, I immediatly ordered the program from S.E. MEDIA. I received the package in about a week and was extremely pleased with the quality of the work that Ray Goff put into this product. And, as did Mickey Ferguson, I readily endorse this product.

However, as was brought out in the review, being a UNIX user, I was forever stumbling over the path syntax. As a result, I have written the following utilities that use the UNIX syntax:

```
CD      Change directory
CATT    Short CATalog list
DLIST   Long Directory list
PWD     Print working directory
```

The syntax that I have used for this uses the forward slash (/) to delimit the directory names and the double dots (..) to designate a parent directory. The drive number can be used by preceding the path with the number and a period.

As an example, to print the directory files for a distant file on a drive other than the working drive, the syntax would be as:

```
LIST 2./CTOOLS/SORTS/QUICK
```

My version of Change Directory replaces both of HIER's SETDIR and CHGDIR utilities. CD entered without a path will seek the root directory of the designated or default drive.

I'm including with this letter the sources for these tools for use by the other "UNIX-type" purchasers of the HIER package. I have also translated the LIST command and I am working on MOVE, COPY and an incremental backup tool all of which work with directory paths regardless of the drive. If anyone is interested in any of these tools, they can drop me a line.

Sincerely,

Bradford Taylor

EDITOR's NOTE: Thanks Brad for the above. We all sure appreciate your sharing with us. I hope to soon place your 'Timestamping & Make Utilities' (similar to Unix/OS-9 makefile utilities) in the S.E. MEDIA catalog (out now for final Beta testing). It makes me proud to see good software and utilities still being offered for FLEX. There are still a lot of FLEX users out there, and we sell FLEX on a regular basis.

*It might be noted in most instances, when we indicate FLEX, SK*DOS applies as well. (Gotcha Pete!)*

Thanks again Brad, keep up the good work.

All these are on 'Reader Service Disk' #31, or contact Brad, either way, if you use HIER (our fastest selling software package) these are a must, especially for the price.

DMW

```
/**********************************************
*                                             *
*  List a directory given a UNIX-like path    *
*                                             *
*  ***DLIST [<drive>.][<path>][<match>]       *
*                                             *
*  path ::= a UNIX like directory path        *
*  match ::= defines the type of files DLIST  *
*            looks at.                        *
*                                             *
*  Example: ***DLIST 2./TOOLS/SOURCES/.TXT    *
*                                             *
*  Prints a vertical directory file list      *
*  similar to the FLEX CATalog utility, given *
*  an optional drive and path list.           *
*  Designed to work with the HIER package     *
*  written by Ray Goff.                       *
*                                             *
*  Bradford Taylor                            *
*  Sharm Engineering                          *
*  Box 97                                     *
*  Mulvane, KS 67110                          *
*  Tel. (316) 777-0706                        *
*                                             *
**********************************************/
#include <stdio.h>
#include <flex.h>

/*  Include Following define only if for use with HIER patch */
/*  Otherwise, comment out or remove                         */

#define HIER  1


#define ROOT 0x0CC5                /*  home track and sector   */
#define UCUCTLG (*mem(0xCC49))     /* user-configurable upper-case flag    */
#define UCO     0x60               /* set for upper-case only              */
#define DELETED 255                /* file deleted code        */
#define SLASH '/'                  /* Path delimit character   */

/*  GLOBALS   */

struct fcb *fp;
char drive;             /*   drive number                  */
char fname[16];         /*   file name area                */
char volname[12];       /*   Volume name taken from SIR    */
int  volnum;            /*   Volume number                 */
unsigned free_size;     /*   number of free sectors        */
char mask[8],m_ext[4];  /*   Match   information           */
char deleteflag;        /*  Print deleted files           */

/*
*** Entry
*/
main(argc,argv)
int argc;
char **argv;
{
      char *path;
      fp = &FLEX_DATA.sysfcb;    /* set pointer to system fcb */

      path = *++argv;
      deleteflag = 0;
      if(argc > 1 && *path == '-')             /* delete option  */
       {
        ++path;
        --argc;
        if(tolower(*path) == 'd')
         deleteflag = 1;                      /*  print deleted files   */
        path = *++argv;
       }

      if(argc < 2)
       path = "\0";
      if(isdigit(*path))
       {
        drive = (*path)&15;
        if(*++path == '.')
         ++path;
       }
      else
       drive = FLEX_DATA . work_drive;   /*  default to work drive */

      open_dir(path);
      show_dir();                        /*  show files in directory */

}

/*
*** open directory from string
*/
open_dir(path)
char *path;
{
      char *pntr;
      unsigned trk_sect,t;        /*  track/sector of file  */

      trk_sect = current_dir();   /*  use current directory */
#ifdef HIER
      if(*path == SLASH)
       {
        trk_sect = ROOT;     /*  start at top  */
        ++path;
       }

      /*  follow path     */
      t = 0;
      if(*path)
       {
        do
         {
          /* check for possible Parent */
          if((strcmp("..",path) ||!strncmp("../",path,3))
           {
            trk_sect = Parent(trk_sect);
            path += 2;
           }
          else
           {
            path = parse_name(path);
            if((t = find_ts(trk_sect)) != -1)
             trk_sect = t;
           }
         }
        while(*path++ == SLASH && t != -1);
        --path;
       }

      if(*path)                  /* proper end of string  */
       not_found();
#else
       path = parse_name(path);
       t = -1;         /* Not HIER Directory Structure  */
#endif

      if(t == -1)       /*  Match information found */
       {
        for(t=0,pntr=fname;t!=8 && *pntr!='.';++pntr,++t)
         mask[t]=*pntr;
        mask[t] = 0;
        while(*pntr && *pntr!='.')
         ++pntr;
        if(*pntr)                     /*  jump over '.'        */
         ++pntr;
        strncpy(m_ext,pntr,3);    /*  copy over extension */
       }
      else
       mask[0]=m_ext[0] = 0;

      /*  get directory name for target directory */
      prints("Drive #");
      flex_outdec(drive,0);
      prints(" Volume: ");
      prints(volnam);
      prints(" #");
      flex_outdec(volnum,0);
#ifdef HIER
      read_sector(trk_sect);     /*read target sector   */
      prints("       Directory: ");
      printn(&(fp->buffer[6]),8);

      /*  Set up FCB for GET_INF  calls  */
      store_int(fp -> buffer,trk_sect);
      fp -> date_index = 0;
      fp -> drive = drive;
#else
      fp -> function = DR_OPEN;
      do_fms();
#endif
      flex_pcrlf();

}

/*
*** parse directory name
*/
parse_name(path)
char *path;
{
        char c,i;

        for(i=0;i!=32 && (c = *path) && c != SLASH;++i,++path)
         {
          fname[i] = UCUCFLG != UCO? c : toupper(c);
         }
        fname[i] = 0;         /* end string */

        return(path);

}

#ifdef HIER
/*
*** Directory not found
*/
not_found()
{
       flex_pstrng("Path name error\4");
       exit();
}

/*
*** Find track sector of directory
*/
find_ts(ts)
unsigned ts;
{

       fp -> drive = drive;        /* Simulate open directory  */
       store_int(fp -> buffer,ts);
       fp -> date_index = 0;

       /*  Find desired file in directory   */
       do
        {
         if(get_info() == -1 || !fp->filename[0])
          return(-1);
        }
       while(strncmp(fp->extension,"DIR",3) ||
             strncmp(fp->filename,fname,8));

       return(to_int(&(fp -> start_track)));

}
#endif

/*
*** Make fms call and parse error
*/
do_fms()
{
      char error;

      error = _fms(fp,'e');     /* call file system */
      if(!(fp->error))
       return(error);
      flex_rpterr(fp);      /*  Report error */
      exit(0);              /*  And die      */

}

/*
*** Return track/sector of current directory
*/
current_dir()
{
       unsigned ts;
       fp -> drive = drive;
       fp -> function = IN_OPEN;  /*  Open SIR  */
       do_fms();

       get_info();
       ts = to_int(&(fp -> buffer[24]));
       free_size = fp -> size;     /* remember free size */
       strncpy(volnam,&(fp->buffer[16]),11))  /*  Get volume name */
       volnum = to_int(&(fp -> buffer[27]));  /*  and volume #     */

       return(ts?ts:ROOT);

}

/*
*** show directory until eof
*/
show_dir()
{

      header();    /* print field labels */

      while(get_info() != -1  && fp -> filename[0])
        {
         if(fp -> filename[0] == DELETED)
          {
           if(deleteflag)
            fp -> filename[0] = '?';
           else
```

```
            continue;
          }
        if(*mask && !amatch(mask,fp->filename))
          continue;
        if(*m_ext && !amatch(m_ext,fp->extension))
          continue;
        flex_pcrlf();
        printn(fp->filename,8);          /*  Name           */
        flex_putchr('.');
        printn(fp->extension,3);         /*  Extension      */
        show_attr(fp->attributes);       /*  Attributes     */
        space();
        flex_outhex(fp->start.track);
        flex_outhex(fp->start.sector);
        flex_putchr('-');
        flex_outhex(fp->end.track);
        flex_outhex(fp->end.sector);
        flex_outdec(fp -> size,1);        /*  display size */
        show_date(&fp->rfu2);             /*  show time of last write */
      }

      flex_pstrng("\nSectors left = \4");
      flex_outdec(free_size,0);

}

/*
*** show attributes of file
*/
show_attr(attr)
char attr;
{

        space();
        if(attr&0x80)
         flex_putchr('W');
        else
         space();
        if(attr&0x40)
         flex_putchr('D');
        else
         space();
        if(attr&0x20)
         flex_putchr('R');
        else
         space();
        if(attr&0x10)
         flex_putchr('C');
        else
         space();

 }

 /*
 *** output write time
 */
 show_date(time)
 char *time;
 {
        int t = time[0];

        flex_outdec(t/10,1);   /*  hour */
        flex_putchr(':');
        t = (t*6)%60;
        flex_putchr(t/10 +'0');
        flex_putchr(t%10 +'0');     /*   minute */
        flex_outdec(time[1],1);
        flex_putchr('-');
        if(time[2]<10)
         flex_putchr('0');
        flex_outdec(time[2],0);
        flex_putchr('-');
        flex_outdec(time[3],0);   /* mon/day/yr  */

 }

 /*
 *** print to either null char or to count
 */
 printn(s,n)
 char *s;
 int n;
 {
        while(*s && n--)
         flex_putchr(*s++);

        if(n > 0)
         spaces(n);        /*  pad rest of field */
 }

 /*
 *** print string
 */
 prints(s)
 char *s;
 {
      while(*s)
       flex_putchr(*s++);

 }
```

```
/*
*** output n spaces
*/
spaces(n)
int n;
{
       while(n--)
        space();

}

space()
{
       flex_putchr(' ');
}

/*
**** match against key
*/
amatch(key,s)
char *key,*s;
{
   while(*key && *key == *s++)
    ++key;
   return(!*key);
}

#ifdef XIER
/*
*** Return track/sector of parent directory
*/
parent(ts)
unsigned ts;
{

       read_sector(ts);          /*  read parent directory     */

       ts =to_int(&(fp->buffer[4]));

       return(!ts?ROOT:ts);      /*  don't backup beyond home  */

}
#endif

/*
*** convert 2 referenced bytes to an integer
*/
to_int(ip)
int *ip;
{
    return(*ip);
}

/*
*** Store integer into a two byte buffer
*/
store_int(ip,i)
int *ip,i;
{
    *ip = i;
}

/*
*** do GET_INF file manager command
*/
get_info()
{
       fp -> function = GET_INF;
       return(do_fms());
}

/*
*** Read a sector
*/
read_sector(ts)
unsigned ts;
{
       fp -> drive = drive;         /* set drive number */
       fp -> function = RD_SEC;     /* set function */
       store_int(&(fp -> current.track),ts);
       return(do_fms());

}

/*
*** Print field labels
*/
header()
{
/*                12345678 EXT DWC  FFFF-TTTT  1234   HH:MM   MM-DD-YY */
  flex_pstrng("NAME      EXT PRT  FROM  TO   SIZE   TIME     DATE\4");
}
```

```
/*********************************************
*                                            *
*  PWD -- PRINT WORKING DIRECTORY            *
*                                            *
*  Report current path list in Unix format   *
*                                            *
*                                            *
*  +++PWD [<drive>]                          *
*                                            *
*  Designed to work with the HIER package    *
*  written by Ray Goff.                      *
*                                            *
*  Bradford Taylor                           *
*  Sharm Engineering                         *
*  Box 97                                    *
*  Mulvane, KS 67110                         *
*  Tel. (316) 777-0706                       *
*                                            *
*********************************************/
#include <stdio.h>
#include <flex.h>

#define ROOT 0x0005       /*  home track and sector   */
#define UCDCTLC (*mem(0xCC49))  /* user-configurable upper-case flag    */
#define UCO     0x60            /* set for upper-case only              */
#define DELETED 255              /* file deleted code        */
#define SLASH '/'                /* Path delimiter           */

/*  GLOBALS   */

struct fcb fblock;      /*   file control block              */
int drive;               /*   drive number                    */

/*
*** Entry
*/
main(argc,argv)
int argc;
char **argv;
{
      char *path;


      if(argc != 2)
       path = "\0";
      else
       path = argv[1];

      if(isdigit(*path))
       {
        drive = (*path++)&15;
        if(*path && *path++ != '.')
         {
          puts("Drive specification error.");
          exit(1);
         }
       }
      else
       drive = FLEX_DATA . work_drive;   /*  default to work drive */

      follow_dir();                 /* follow path and display  *.

}

/*
*** follow backward directory path
*/
follow_dir()
{
      unsigned trk_sect;      /*  track/sector of file  */

      trk_sect = current_dir(&fblock);     *  use current directory */

      putchar(drive+'0');        /* report drive first */
      putchar('.');
      if(trk_sect == ROOT)
       putchar(SLASH);
      else
       report_path(trk_sect);          /* report path  */
}

/*
*** Report Path
*/
report_path(start)
unsigned start;
{
     char name[16];

     if(start != ROOT)
      {
       start=parent(&fblock,start,name);
       report_path(start);
       putchar(SLASH);
       puts(name);
      }
}

/*
```

```
*** Make fms call and parse error
*/
do_fms(fp)
struct fcb *fp;
{
      int error;

      error = _fms(fp,'a');     /* call file system */
      if(fp->error)
       {
        flex_rpterr(fp);
        exit(0);
       }
      return(error);

}

/*
*** Return track/sector of current directory
*/
current_dir(fp)
struct fcb *fp;
{
       fp -> drive = drive;
       fp -> function = IN_OPEN;  /*  Open DIR  */
       do_fms(fp);

       fp -> function = GET_INF;  /*  Get information */
       do_fms(fp);

       return(to_int(&(fp -> buffer[24])));

}

/*
*** Return track/sector of parent directory
*/
parent(fp,ts,name)
struct fcb *fp;
unsigned ts;
char *name;
{
      fp->drive = drive;
      fp->current.track = ts>>8;
      fp->current.sector = ts&255;

      fp->function = RD_SEC;
      do_fms(fp);                /* get that information     */

      ts =to_int(&(fp->buffer[4]));

      strncpy(name,&(fp->buffer[6]),8);

      return(!ts?ROOT:ts);     /*  don't backup beyond home  */

}

/*
*** convert 2 pointed to bytes to an integer
*/
to_int(ip)
int *ip;
{
   return(*ip);
}
/*********************************************
*                                            *
*  Change directory using UNIX format        *
*                                            *
*  General syntax:                           *
*                                            *
*  +++CD [<drive>.][<path>]                  *
*                                            *
*  drive ::=  the optional disk drive number *
*             to set the current directory to *
*                                            *
*  path  ::=  a UNIX-like directory path     *
*                                            *
*  CD without parameters or only a drive #   *
*  will change to the home directory of the  *
*  work or designated drive.                 *
*                                            *
* Example:                                   *
*         +++CD ../TOOLS/SOURCES             *
*                                            *
*  CD was designed to work with the HIER     *
*  package by Ray Goff.                      *
*                                            *
*  Bradford Taylor                           *
*  Sharm Engineering                         *
*  Box 97                                    *
*  Mulvane, Ks. 67110                        *
*  Tel. (316) 777-0706                       *
*                                            *
*********************************************/
```

```
#include <stdio.h>
#include <flex.h>

#define BOOT 0x0003        /*  home track and sector   */
#define [illegible] (*mem(0xCC49))  /* user-configurable upper-case flag     */
#define UCO      0x60             /* set for upper-case only               */
#define SLASH '/'                 /* slash delimiter              */

/*  GLOBALS   */

struct fcb fblock;       /*   file control block              */
struct fcb *fp;          /*   pointer to file control block   */
int drive;               /*   drive number                    */
char fname[16];          /*   file name area                  */

/*
*** Entry
*/
main(argc,argv)
int argc;
char **argv;
{
        char *path;

        fp = &fblock;          /*  set global pointer  */

        if(argc != 2)
         path = "/";
        else
         path = argv[1];

        if(isdigit(*path))
         {
          drive = (*path++)&15;
          if(*path++ != '.')
           {
            puts("Drive specification error");
            exit();
           }
         }
        else
         drive = FLEX_DATA . work_drive;   /*  default to work drive */

        change_dir(path);
```

To Be Continued Next Month

Classifieds *As submitted - No Guarantees - As is!*

DAISY WHEEL PRINTERS
Qume Sprint 9 - $900 Qume Sprint 5 - $800

Winchester 10 Megabyte Drive - Seagate Model #412 $275.

3 - Dual 8" drive enclosure with power supply. New in box. $125 each.

5 - Siemens 8" Disk Drives. $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS $495.

TELETYPE Model 43 PRINTER - with serial (RS232) interface and full ASCII keyboard. $250 ready to run.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09 CU Card - $900 complete.

CDS-1- 20 Meg Hard Disk System with controller $500.

(615) 842-4600 M-F 9 AM to 5 PM EST

***

EXORBUS Memory Boards, several 32K battery backed STATIC RAM memory cards. Call (214) 986-1220 Dave C.

SK★DOS™

The Generic DOS™ for 68000 applications in

★ Industrial Control
★ Business Use
★ Educational Computing
★ Scientific Computing
★ Number Crunching
★ Dedicated Systems
★ Turnkey Systems
★ Data Collection
★ Single-board Computers
★ Bus-oriented Computers
★ Graphics Workstations
★ One-of-a-kind Systems
★ Advanced Hobbyist Use

SK★DOS is a single-user disk operating system for computers using Motorola 32 bit CPUs such as the 68008, 68000, 68010, and 68020. It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes. Because SK★DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK★DOS comes with over 40 commands and system programs, including a 6809 emulator which allows 68K **SK★DOS** to run application programs and languages developed for 6809 **SK★DOS** and other systems. Assemblers, editors, and higher level language support are available from third party software vendors and through public domain software.

SK★DOS is available for single-copy or dealer sales, as well as OEM licensing. Single copies cost $125 (inquire as to available systems). Extremely attractive OEM licensing terms are also available. An optional Configuration Kit contains a detailed Configuration Manual and two disks of source code for system adaptation, including source code for a system monitor/debug ROM and other programs useful for adapting SK★DOS to new systems.

SK★DOS

is available from

STAR-K
SOFTWARE SYSTEMS CORPORATION
P.O. BOX 209 · MT. KISCO, NY 10549 · 914/241-0287
TELEX 5106016774

Call for Software:

68000, C, Basic09 Sculptor

We are receiving calls and letters from numerous sources, including users, business and others looking for OS-9 68000 software: applications, etc.

Many of you have developed software that with little change could be adapted for others. If you are interesting in selling it, please let us know. There is a growing market out there now. Get in on the ground floor!

If you can use additional income and have something that might be of interest, call and talk to Larry or Don.

S.E. MEDIA Division - CPI
POB 849
Hixson, TN 37343

Telephone (615) 842-6809
Telex (510) 600-6630

Now! *"TOPS The OFFICE PRINT Shop™"* Makes *professionals* of us all. For *less* than just affordable!

DeskTop Publishing

NOTE: The following includes a 3 day, *hands-on* , instructional session for the entire *"The OFFICE PRINT Shop™"* system. A full 6 months, *no extra charge*, telephone or in-house (our offices, normal business hours) follow-up advisory service. We feature full Apple™ service and also national service by Honeywell. This includes *next day* , on site service. Over 95% of all service requests are completed on the initial call. ***We understand the importance of fast service!***

100 **System 100** *Very Affordable* *Save up to 90% on your printing*

The *TOPS* System 100
consists of the following items:
- A special Apple Macintosh Plus™ 1 Megabyte computer, including a double sided, double density 800,000+ character disk drive.
An Apple LaserWriter™ *typeset quality* printer.

Page make-up software
galley typesetting software
3 day - on site - instructions
6 months instructional support &
much more!

Option:
We also offer software to drive *most all* the large commercial typesetters. You can save a bundle by doing your own typesetting and proofing, and then downloading to the commercial system.

Leasing with payout available for all systems.

Also available with 20 million character storage 'Hard Disk'

Data-Comp Division

CPI A Decade of Quality Service' *Systems World-Wide*

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

This document *composed, typeset and printed* with a TOPS System 100

TOPS The OFFICE PRINT Shop is a trademark of Computer Publishing, Inc.
Apple Macintosh Plus is a trademark of Apple Computer Company, Inc.
LaserWriter is a trademark of Apple Computer Company, Inc.

OS-9 UniFLEX MUSTANG-020, 68020, 68881 AND MORE HANDS-ON EXPERIENCE

The DATA-Comp Division of Computer Publishing Corporation announces their new and innovative HANDS-ON 68020 computer familiarization two day event. A chance to TRY BEFORE YOU BUY!

For two full days (Monday through Friday - excluding legal holidays) each participant will be furnished the exclusive use of a 68020 computer (MUSTANG-020). Each system will have available native C compilers, BASIC, assembler and other high level languages. Each system will be equipped with the Motorola MC 68881 math co-processor, where applicable.

Each demonstration room will contain not more than two work stations. Each system will be equipped with floppy disk, 20 megabyte winchester technology hard disk, and 2 megabyte of RAM. RAM is partitioned as 690K bytes of RAM disk and 1.2 megabyte of user RAM space.

Participants are encouraged to bring along any source level projects, for evaluation, in C, BASIC or assembler. Call for availibility of other HHLs.

Although this is not a training seminar, Data-Comp personnel are available for assistance and consultation. This event is scheduled for hands-on evaluations of the 68020 CPU, 68881 math co-processor and MUSTANG-020 system, operating in a functional environment.

Transportation to and from the airport and hotel/motel will be provided. Lunch provided both days. Chattanooga airport is serviced by American, Delta, Republic and other airlines.

COST

One person - $375.00

Two persons - $595.00

* Motel single $22.00, double $26.00
Includes satellite TV - convenient to food and shopping

DATA-COMP

A Division of Computer Publishing, Inc.

5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

Systems available for both OS-9 and UniFLEX. Reservation should be made 15 days in advance. Attendee should initially indicate OS-9, UniFLEX or both. Special facilities available on request. Please write or call for additional information.

NOTE: Both OS-9 and UniFLEX are Unix type operating systems. Each as been enhanced in some aspect or another. Prospective attendees should have some working knowledge or experience with one of these operating systems, to gain full benefit of the session. However, a newcomer will find that it is a simple matter to be fairly proficient in using these systems in the allocated time. Special system instruction available on request. Call or write.

* Hotel/Motel cost are separate cost, not included in the basic cost shown.

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk. OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINTC4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1, .C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
* All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly

VISA

**Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343**

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

6805
Development Systems
for
MS-DOS Computers

Use the IBM PC/XT/AT or compatible MS-DOS computers to develop applications based on Motorola's 6805 single chip microcomputers. These complete intergrated systems consist of a cross assembler program, a full screen simulator/debugger program and one programming circuit board with menu driven driver software. The model MCPM-1 prog board supports the MC68705P3, P5, U3, U5, R3 and R5 nmos processors while the model MCPM-2 board supports the MC1468705F2 & G2 cmos versions. The programming boards connect to the computer via a serial port and allow skipping the usual eprom programming step when used in a development environment, however, an on-board eprom programmer is provided so that the boards can be used in a stand alone mode. The system components are avail separately.

Complete Sys. specify MCPM-/2 ....$495
Shipping - add 3% for U.S.A. and Canada

TEC
P.O. Box 53 West Glover, Vermont 05875
Phone (802) 525-3458

SPECIALTY ELECTRONICS
S-O-F-T-W-A-R-E
for
* **OS9/68000**

ACCOUNTING	
Accounts Receivable	$399
Accounts Payable..................	$399
General Ledger/Cash Journal	$499
Payroll	$499
Inventory Control	$499
UTILITIES	
Disk Repair	$79.95
Modem 68K	$79.95
Chgattr............................	$19.95
Dirs	$19.95

For more information contact
Specialty Electronics
909 N. Cleveland / Enid, Oklahoma 73703
(405) 233-1632

* trademarks OS9 Microware, Inc.

Lloyd I/O is a computer engineering corporation providing software and hardware products and consulting services.

19535 NE GLISAN • PORTLAND, OR 97230 (USA)
PHONE: (503) 666-1097 • TELEX: 910 380 5448 LLOYD I O

Computer Engineers

K-BASIC™ *IS HERE*
K-BASIC is a TSC XBASIC (XPC) compatible COMPILER
for OS9 & FLEX... price $199

Here at last is a compiler for BASIC that will compile all your XBASIC programs. K-BASIC compiles TSC's XBASIC and XPC programs to machine code. K-BASIC is ready now to save you money and time by teaching your computer to:
• Think Faster • Conserve Memory • Be Friendlier

Call (503) 666-1097 for our CATALOG.
We have many programs for serious software developers!

DO™
Micro BASIC for OS9... $149

A structured mirco BASIC for general system control featuring: Parameter passing, 10 string variables, 26 numeric variables, subroutines, nested loops, interactive I/O, sequential files, and time variables (for applications executing in the background required to execute procedures such as disk or file backups.) Includes the SEARCH and RESCUE UTILITIES™. (For OS9 ONLY.)

SEARCH and RESCUE UTILITIES™
for OS9... $35

A super directory search utility. Output may be piped to the included utilities to perform file: COPIES, DELETES, MOVES, LISTING (pagination), and FILTERING. Some filtering utility programs are included; of interest is the FILE DATE CHECKING utilities: YOUNGER and DRAFT (Level 2). (For OS9 Level 1 and 2.)

PATCH™
Modem Communications for OS9... $39

PATCH is a modem communications program for OS9 featuring: KEY MACROS, ASCII TEXT AND BINARY FILE UP/DOWN LOADING, PRINTER COPY, and HELP MENUS. We use it several times each day with our TELEX service. PATCH is convenient and easy to use. Key macros may be pre-stored and loaded at any time.

CRASMB™
CROSS ASSEMBLER PACKAGE
for OS9 & FLEX... all for $399
Motorola CPU's... $150
Intel CPU's... $150, Others... $150

CRASMB is the highly acclaimed cros[illegible]ge for OS9 and FLEX systems. It [illegible] your [illegible] a development statio[illegible] get C[illegible]:
[illegible] 68[illegible] 805 6809 6811 [illegible]02
[illegible] 8048 8051 80[illegible] Z80
(68000 [illegible] ...$249)

NOW FOR OS9/68K $432

CRASMB features [illegible] Conditi[illegible] Long symbol names, Symbol cross reference tab[illegible] code in 4 formats (OS9, FLEX, S-1-S9, INTEL HEX).

VISA, MC, COD, CHECKS, ACCEPTED

USA:	LLOYD I/O (503 666-1 97),	S.E. MEDIA (800 338 6800)
England:	Vivaway (0582 423425),	Windrush (0692 405189)
Germany:	Zacher Computer (65 25 299),	Kell Software (06203 6741)
Australia:	Paris Radio Electronics (344 9111)	
Japan:	Microboards (0474) 22-1741	Seikou (03) 832-6000
Switzerland:	Elsoft AG (056 86 27 24)	
Sweden:	Micromaster Scandinavian AG (018 - 138595)	

K-BASIC, DO, SEARCH and RESCUE UTILITIES, PATCH, CRASMB and CRASMB 16.32 are trademarks of LLOYD I/O
OS9 is a ™ of Microware, FLEX is a ™ of TSC

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS/9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
NE : 68010 disassembler $100-FLEX,OS/9,UNIFLEX,OS/9-68K,MSDOS

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS/9,UNIFLEX,MSDOS ANY 3 $100 ALL $200
specify for 180x,6502,6801,6804,6805,6809,Z8,Z80,8048,8051,8085,68000
modular cross-assemblers in C, with load/unload utilities NOW: OS/9-68K
8-bit (not 68000) sources for additional $50 each, $100 for 3, $300 for all

DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS/9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50 COCO FLEX,COCO OS/9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS/9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809	$75-FLEX $85-OS/9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind.	$50-FLEX $75-OS/9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequence only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS/9,UNIFLEX,MS-DOS,OS/9-68K,UNIX
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $12.50-SSSD/SSDD/DSDD
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-4570 or 1717

We take orders at any time, but plan long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware; COCO Tandy; MSDOS Microsoft.

SOFTWARE FOR THE HARDCORE

•• FORTH PROGRAMMING TOOLS from the 68XX&X ••
•• FORTH specialists — get the best!! ••

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET.COMPILATION for

6800, 6301/6801, 6809, 68000, 8080, Z80

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than compiled languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH™ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

™ tFORTH and firmFORTH are trademarks of Talbot Microsystems
™ FLEX is a trademark of Technical Systems Consultants, Inc
™ CP/M-68K is trademark of Digital Research, Inc

tFORTH™
from TALBOT MICROSYSTEMS
NEW SYSTEMS FOR
6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORcisor Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

•• tFORTH — extended fig FORTH (1 disk) $100 ($15)
with fig line editor.
•• tFORTH + — more! (3 5" or 2 8" disks) $250 ($25)
adds screen editor, assembler, extended data types, utilities, games, and debugging aids.
•• TRS-80 COLORFORTH — available from The Micro Works
•• firm FORTH — 6809 only. $350 ($10)
For target compilations to rommable code.
Automatically deletes unused code. Includes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH +.
•• FORTH PROGRAMMING AIDS — elaborate decompiler $150

•• tFORTH for HX-20, in 16K roms for expansion unit or replace BASIC $170

•• tFORTH/68K for CP/M-68K 8" disk system $290
Makes Model 16 a super software development system.

•• Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) $200
— TARGET source code: 6800-$200, 6301/6801—$200
same plus HX-20 extensions— $300
6809—$300, 8080/Z80—$200, 68000—$350

Manuals available separately — price in ( ).
Add $6 system for shipping, $15 for foreign air.

TALBOT MICROSYSTEMS 1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

PLµS-68k (PL/9 for the 68000) running under OS/9-68000

- Built-in screen editor
- Built-in source-level debugger
- Byte, Integer, Long and Real variables
- Direct source to object compilation
- Generates fast, efficient object code
- Compiles over 2000 lines/min
- 99% upward compatible with PL/9
- Comprehensive instruction manual
- No license fees to pay

PLµS-68k is a complete re-write of PL/9, for the 68000-series µP. It is a structured high-level language, designed for programming stand-alone applications as well as those in which an operating system is present. It is easy to learn, very quick to use (the compiler is resident with the editor and produces executable object code in one pass) and generates efficient, position-independant code upon which no royalties are payable.

The built-in tracer/debugger helps you debug your programs symbolically and at source level, so you need know next to nothing about the assembly-language of the 68000 in order to develop programs.

PLµS is here now, it carries the Windrush commitment to support, and it's cheap. Do you need a better reason to try it?

PLµS-68k for OS/9-68000............................$299.00

PLµS-68k is also available for FLEX™ systems, supplied with a 68008 second processor, case and power supply. (This reduced price no longer includes a free upgrade to the OS/9 version)................$499.00

For further information, phone or write:

Worstead Laboratories
North Walsham
Norfolk NR28 9SA
England

WINDRUSH
Micro Systems Ltd.

Tel (44) 692 404086
Telex 975548 WMICRO G

'68' MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card # ______________________ Exp. Date _____

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $ __________

Name ______________________

Street ______________________

City ______________ State _____ Zip _______

My Computer Is: ______________

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

68 Micro Journal
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600

Telex 510 600-6630

CSG IMS

CSG IMS is a general purpose information management system designed to make the development of file-intensive applications as quick and easy as possible. IMS is a full featured database manager with the added benefit of a structured, general purpose application language. Some popular applications are: accounting, inventory, data acquisition, cataloging, membership and mailing lists.

SYSTEM FEATURES

- CSG IMS uses B+Tree index structures for fast database access and reliability. Record, index and file sizes are virtually unrestricted.
- Supported data types are: text, BCD floating point (14 digits), short and long integers, and date.
- Menu driven executive program for ease of operation.
- User definable screen forms and reports are supported.
- The interactive environment provides access to databases and most language features allowing quick ad hoc queries.
- CSG IMS includes a recursive, compiled language supporting program modules with full parameter passing.
- The CSG IMS run-time interpreter is available separately for user developed and distributed applications.
- Comprehensive 320 page manual with tutorial section.

CSG IMS for OS9/6809 LII and OS9/68000: $495.00
Run time interpreter for CSG IMS: $100.00
CSG IMS manual only: $20.00
PRICES IN US DOLLARS
ADD $5.00 S&H FOR CONTINENTAL USA, FOREIGN ORDERS ADD $10.00 S&H

Clearbrook Software Group

To order CSG IMS or to receive further information write:
CLEARBROOK SOFTWARE GROUP
P.O. Box 8000-499
Sumas, WA 98295-8000
or phone:
(604) 853-9118

Send for a free catalog describing all of our OS9 products.
We welcome dealer inquiries.

OS9 is a registered trademark of Microware and Motorola.

OS-9™ SOFTWARE

L1 UTILITY PAK—Contains all programs formerly in Filter kits 1 & 2, and Hacker's kit 1 plus several additional programs. Complete "wild card" file operations, copies, moves, sorts, del, MACGEN shell command language compiler, Disassembler, Disk sector edit utility, new and improved editions, approx. 40 programs, increases your productivity. Most programs applicable for both level I & II 6809 OS-9. **$49.95 ($51.95)**

Call or send Self Addressed Stamped Envelope for catalog of software for color Computer OS-9 and other OS-9 systems.

BOLD prices are CoCo OS-9 format disk, other formats (In parenthesis) specify format. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

SS-50C
MEMORY LIQUIDATION SALE!
(While Supply Lasts)

1 MEGABYTE RAM BOARD
Full megabyte of ram with disable options to suit any SS-50 6809 system. High reliability, can replace static ram for fraction of the cost. $399 for 2 Mhz or $439 for 2.25 Mhz board assembled, tested and fully populated.

2 MEGABYTE RAM DISK BOARD
RD2 2 megabytes dedicated ram disk board for SS-50 systems. Four layer circuit board socketed for 2 **Megabytes!** Special sale price of **$399.00** includes only 256k of ram installed (you add the rest), includes OS-9 level I and II drivers for Ram disk, (note: you can re-boot your system without losing ram-disk contents). (Add $6 shipping and insurance.)

Please call for answers to your technical questions concerning these products.

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223, (503) 244-8152
(For best service call between 9-11 am Pacific time.)

OS-9 is a trademark of Microwave and Motorola Inc.
MS-DOS is a trademark of Microsoft Inc.

COMPILER EVALUATION SERVICES

BY: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc.
is offering the following SUBSCRIBER SERVICE:

COMPILER COMPARISON AND EVALUATION REPORT

Due to the constant and rapid updating and enhancement of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes of does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author.

PASCAL 'C' GSPL WHIMSICAL PL/9

Initial Subscription - $39.95
(includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - C.P.I.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

68000 68020 68010

68008 6809 6800

Write or phone for catalog.

AAA Chicago Computer Center

120 Chestnut Lane — Wheeling IL 60090
(312) 459-0450

Technical Consultation available most weekdays from 4 PM to 6 PM CST

Stop!

Get a 25 MegaByte Hard Disk practically FREE - only 1¢

Be Sure to Consider the SPECIAL MUSTANG 1¢ Sale on page 5

When it's over, IT'S OVER!

We don't know how long this very, very low price can be maintained, don't miss it!

Data-Comp Div. - CPI

6809<>68XXX UniFLEX

X-TALK

A C-MODEM/Hardware Hookup

Exclusive for the MUSTANG-020 running UniFLEX, is a new transfer program and cable set from DATA-COMP (CPI). X-TALK consist of 2 disks and a special cable, this hook-up enables a 6809 SWTPC UniFLEX computer to port UniFLEX files directly to a 68XXX UniFLEX system.

This is the only currently available method to transfer files, text or otherwise, from a 6809 UniFLEX system to a 68000 UniFLEX system, that we have seen. A must if you want to recompile or cross assemble your old (and valuable) source files to run on a 68000 UniFLEX system. GIMIX users can directly transfer files between a 6809 GIMIX system and our MUSTANG-020 68020 system, or GIMIX 68020 system. All SWTPC users must use some sort of method other than direct disk transfer. The 6809 SWTPC UniFLEX disk format is not readable by most other 68000 type systems.

The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 DB25 connectors. A special SWPTC+ cable and software is also available, at the same price. Orders must specify which type SWTPC 6809 UniFLEX system they intend to transfer from or to.

The X-TALK software is furnished on two disks. One 8" disk containing the 6809 software and one 5" disk containing the 68XXX software. These programs are also complete MODEM programs and can be used as such, including X-on X-off, and all the other features you would expect from a full modem program.

X-TALK can be purchased with/without the special cables, however, this SPECIAL price is available only to registered MUSTANG-020 owners.

X-TALK, w/cable	**$ 99.95**
X-TALK only	**69.95**
X-TALK w/source	**$149.95**

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

Telephone 615 842-4601
Telex 510 600-6630

Note: Registered MUSTANG-020 owners must furnish system serial number in order to buy at these special low prices.

68 MICRO JOURNAL Reader Service Disks

Disk- 1	Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet
Disk- 2	Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn, **Lifetime
Disk- 3	Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
Disk- 4	Mailing Program, *Finddat, *Change, *Testdisk
Disk- 5	*DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk- 6	**Purchase Order, Index (Disk file indx).
Disk- 7	Linking Loader, Rload, Harkness.
Disk- 8	Crtest, Lanpher (May 82).
Disk- 9	Datecopy, Diskfix9 (Aug 82).
Disk-10	Home Accounting (July 82).
Disk-11	Dissembler (June 84).
Disk-12	Modem68 (May 84).
Disk-13	*Initmf68, Testmf68, *Cleanup, *Diskalign, Help, Date.Txt
Disk-14	*Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib.
Disk-15	Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16	Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17	Match Utility, RATBAS, A Basic Preprocessor.
Disk-18	Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19	Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20	UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C
Disk-21	Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22	Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23	ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March1986.
Disk-24	68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25	KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26	Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27	ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28	CT-82 Emulator, bit mapped.
Disk-29	**Star Trek
Disk-30	Simple Winchester, Dec.'86 Green.
Disk-31	*** Read/Write MS/PC-DOS (SK*DOS)

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

Specify 8" disk $19.50
5" disk $16.95

Add: S/H - $3.50
Overseas add: $4.50 surface - $7.00 Air Mail, USA Dollars

68 MICRO JOURNAL
PO Box 849
Hixson, TN 37343
615 842-4600 - Telex 510 600-6630

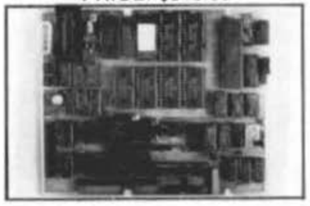
6809/68008 SINGLE BOARD COMPUTERS

The Peripheral Technology Family of Single Board Computers is a Low-Cost Group Which Ranges From an Entry Level 8-Bit Version to a Powerful 68008-Based Board. A Product is Available to Fit Almost Every User's Requirements.

PT69-5

- 6809 Processor/2MHZ Clock
- 4 RS-232 Serial Ports
- 2 8-Bit Parallel Ports
- 4K-16K EPROM/60K Ram
- Parallel Printer Interface
- DS/DD Controller for 35-80
- Track Drives Ranging From SS/SD-DS/DD
- Winchester Interface Port

PRICE: $349.00



PT69-3

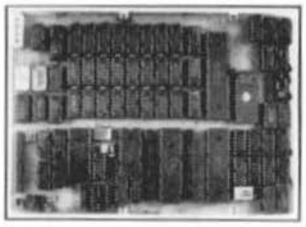
- 6809 1 MHZ Processor
- 2 RS-232 Serial Ports
- 2 8-Bit Parallel Ports
- 4K EPROM/59K User Ram
- DS/DD Controller for 35-80 Track Drives Ranging From SS/SD-DS/DD

PRICE: $269.95
OS9 L1 For
PT69 BOARDS: $200.00
SK*DOS: $ 49.95

PT68K-1

- MC68008 10 MHZ Processor
- 768K RAM/64K EPROM
- 2 RS-232 Serial Ports
- Winchester Interface Port
- Floppy Disk Controller for 2 5+" Drives
- 2 8-Bit Parallel Ports

BOARD: $595.00
WITH OS9: $749.95
WITH SK*DOS: $675.00



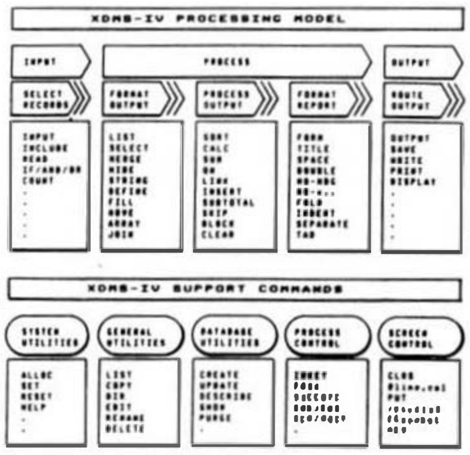
PERIPHERAL TECHNOLOGY
1480 Terrell Mill Road, Suite 870
Marietta, Georgia 30067
(404) 984-0742 Telex # 880584
VISA/MASTERCARD/CHECK/C.O.D.

™OS9 Is A Trademark Of Microware and Motorola.

Send For Catalogue For Complete Information On All Products.

XDMS-IV

Data Management System



Save $100.00 - Limited Time
Regular $350.00 - Now Only
$249.95

Technical telephone assistance: Tel 914-941-3552 (Evenings)
FLEX™ Tecnical Systems Consultants, SK*DOS™ STAR-KITS Corp.

FOR 6809 FLEX-SK-DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character filed names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV **is a brand new approach** to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...). The possibilities are unlimited.

Visa & Master Card Excepted

Telephone: 615-842-4601 or Telex: 510 600-6630
Or Write: S.E. Media, 5900 Cassandra Smith Rd., Hixson, Tenn. 37343

GMX Micro-20 prices

MICRO 20 (12.5 MHz)	$2565.00
MICRO 20 (16.67 MHz)	$2895.00
8 PORT RS232 BOARD SET (SBC-BS)	$ 498.00
PROTOTYPING BOARD (SBC-WW)	$ 75.00
BACK PANEL PLATE (BPP-PC)	$ 44.00
I/O BUS ADAPTER (SBC-BA)	$ 195.00

QUANTITY DISCOUNTS ARE AVAILABLE ON THE ABOVE ITEMS AS FOLLOWS: 4-9, LESS 5%; 10-24, LESS 10%; 25-99, LESS 20%; 100 UP, LESS 30%.

MC68881RC12	$ 295.00
MC68881RC16	$ 395.00
SBC ACCESSORY PACKAGE (M20-AP)	$1690.00

For other configurations and options, contact GMX.

MOTOROLA 68020 USERS MANUAL	$ 10.00
MOTOROLA 68881 USERS MANUAL	$ 10.00

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GMX

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609

(312) 927-5510 • TWX 910-221-4055

GMX S-50 BUS prices

68020 SYSTEM

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer, DMA transfers, high speed MMU, we have the UNIFLEX-VM 68020 development system.
The system CPU provides protection to the system and other users from crashes caused by defective user programs.
The system's intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions.
The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. It allows up to 4 Megabytes of Virtual Memory per user. All systems include 1MB of static RAM, one 3-port Intelligent Serial I/O board, DMA Controllers, a 5" 80 track floppy drive.

PRICES

#020 UniFLEX VM with 25MB HD	$10,980.20
#020 UniFLEX VM with 85MB HD	$12,480.20

YOU CAN EXPAND THESE 020 SYSTEMS WITH:

60MB STREAMER	$ 2,400.00
REMOVABLE PACK DRIVE	$ 1,200.00

INTELLIGENT I/OS

#14 3 Port Serial-30 Pin	$	498.14
#13 4 Port Serial-50 Pin	$	618.13
#12 Parallel-50 Pin	$	538.12

CABLE SETS FOR I/OS

# 95 Cable Sets Specify Card	$	24.95
# 51 Cent. B.P. Cable for #12 & #44	$	34.51
# 53 Cent. Cable Set	$	36.53

6809 SYSTEM

The number 39 systems include: #05 CPUwDAT; #19 Classy Chassis; 256K Static RAM; a # 43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates;

System # 39 OS-9 GMX II Dual 80 DSDD	$ 2,998.39
" w19MB	$ 4,698.39
w72MB	$ 6,298.39

The Software included in this System:
GMXBUG monitor; FLEX; and OS-9 GMXII. You can software select either FLEX or OS-9. Also includes OS-9 Editor, Assembler, Debugger, BASIC-09, RUNB, RMS, DO, and GMX-VDISK for FLEX.

System # 39 UniFLEX w25MB	$ 4,698.39
" w85MB	$ 6,298.39

The UniFLEX Operating System is included.

6809 SYSTEMS USING THE GIMIX III CPU & INTELLIGENT I/O PROCESSOR BOARDS

These System include: GMX6809 CPU III; one #11 3 port intelligent serial I/O & Cables; #19 Classy Chassis; 256K Static RAM; #68 DMA controller; all necessary cables, power regulators, and filler plates.

System # 79 OS9 GMX III Dual 80 DSDD	$ 4,498.79
" w25MB	$ 6,498.79
" w85MB	$ 7,998.79

The # 79 System Software includes: OS9 GMXIII; OS9 Editor, Assembler, Debugger, BASIC 09, RUNB, RMS, DO, RAMdisk, O-FLEX; GMXBUG; FLEX. The GMX Support ROM and the hardware CRC board are exclusive features included in this system.

System # 89 UniFLEX III w25MB	$ 6,798.39
w85MB	$ 8,298.39

The UniFLEX GMX III Operating System is included.

For Ordering Call
(615)842-4600

FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS, RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Went J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

Introducing ....

S - 50 BUS / 68XX

Board and/or Computer
Terminals-CRTs-Printers
Disk Drives-etc.

REPAIRS

This

NOW AVAILABLE TO ALL S50/68XX USERS

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to **'ALL'** S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SSB - HELIX and others, including the single board computers. *Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

1. If you require service, the first thing you need to do is call the number below and describe your problem and *confirm a Data-Comp service & shipping number!* This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

2. All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

3. If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates *must be requested.* Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with the item to be serviced. Please note that estimates are given in dollar amounts only.

4. Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

Not This

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615)842-4607
Telex 5106006630